bib2gls: a command line Java application to convert .bib files to glossaries-extra.sty resource files

Nicola Talbot
dickimaw-books.com

Version 3.4 2023-06-29

The bib2gls command line application can be used to extract glossary information stored in a .bib file and convert it into glossary entry definitions that can be read using glossaries-extra’s \GlsXtrLoadResources command. When used in combination with the record package option, bib2gls can select only those entries that have been used in the document, as well as any dependent entries, which reduces the \TeX resources required by not defining unnecessary commands.

Since bib2gls can also sort and collate the recorded locations present in the .aux file, it can simultaneously by-pass the need to use makeindex or xindy, although bib2gls can be used together with an external indexing application if required. (For example, if a custom xindy rule is needed.)

An additional build may be required to ensure the locations are up-to-date as the page-breaking may be slightly different on the first \TeX run due to the unknown references being replaced with ?? which can be significantly shorter than the actual text produced when the reference is known.

Note that bib2gls is a Java application, and requires at least Java 8.\footnote{The List.sort method used to sort the entries was only introduced to Java 8.} Additionally, glossaries-extra must be at least version 1.12. (Although the latest version is recommended.) This application was developed in response to the question “Is there a program for managing glossary tags?” on \TeX on StackExchange [17]. The .bib file can be managed in an application such as JabRef.

If you already have a .tex file containing entry definitions using commands like \newglossaryentry then you can use the supplementary tool convert-gls2bib to convert the entries to the .bib format required by bib2gls. See chapter 7 for further details.

1 The List.sort method used to sort the entries was only introduced to Java 8.
The supplementary file “glossaries-extra and bib2gls: An Introductory Guide” (bib2gls-begin.pdf) is an introductory guide to the glossaries-extra package, which you may prefer to start with if you are unfamiliar with the glossaries and glossaries-extra packages.

Additional resources:

• bib2gls gallery.
• bib2gls FAQ

TUGboat articles:

• Glossaries with bib2gls, issue 40:1, 2019.
• bib2gls: selection, cross-references and locations, issue 41:3, 2020.
• bib2gls: sorting, issue 42:2, 2021.
# Contents

Glossary .......................... xx

1 Introduction .......................... 1
   1.1 Example Use .......................... 2
   1.2 Logical Divisions: type vs group vs parent .......................... 4
   1.3 Defining a New Glossary .......................... 10
   1.4 Resource Sets .......................... 11
   1.5 bib2gls Quarks .......................... 15
   1.6 Indexing .......................... 16
   1.7 Security .......................... 17
   1.8 Localisation .......................... 18
   1.9 Conditional Document Build .......................... 18
   1.10 Manual Installation .......................... 19

2 \TeX{} Parser Library .......................... 21

3 Command Line Options .................................. 29
   3.1 Common Options .................................. 29
      --help (or -h) .................................. 29
      --version (or -v) .......................... 29
      --debug \[n\] .......................... 29
      --no-debug (or --nodebug) .......................... 29
      --verbose .......................... 29
      --no-verbose (or --noverbose) .......................... 29
      --quiet (or -q) .......................... 30
      --silent .......................... 30
      --locale \[lang\] (or \-l \[lang\]) .......................... 30
      --group (or \-g) .......................... 30
      --no-group .......................... 34
   3.2 File Options .................................. 34
      --dir \langle dirname\rangle (or \-d \langle dirname\rangle) .......................... 34
      --log-file \langle filename\rangle (or \-t \langle filename\rangle) .......................... 35
      --tex-encoding \langle name\rangle .......................... 35
   3.3 Interpreter Options .................................. 35
      --break-space .......................... 35
      --no-break-space .......................... 36
      --custom-packages \langle list\rangle .......................... 36
--ignore-packages ⟨list⟩ (or -k ⟨list⟩) ........................................ 36
--interpret ................................................................. 36
--no-interpret .............................................................. 36
--list-known-packages ..................................................... 36
--packages ⟨list⟩ (or -p ⟨list⟩) ............................................ 37
--support-unicode-script .................................................. 37
--no-support-unicode-script .............................................. 38
--obey-aux-catcode ....................................................... 38
--no-obey-aux-catcode .................................................... 38

3.4 Record Options .......................................................... 38
--cite-as-record ............................................................ 38
--no-cite-as-record ........................................................ 38
--collapse-same-location-range .......................................... 38
--no-collapse-same-location-range ..................................... 39
--merge-namerefn ⟨rule⟩ .................................................... 38
--no-merge-namerefn ⟨rule⟩ ............................................... 38
--record-count ⟨or -c⟩ ...................................................... 42
--no-record-count .......................................................... 42
--record-count-unit ⟨or -n⟩ ................................................ 43
--no-record-count-unit .................................................... 43
--record-count-rule ⟨rule⟩ ................................................ 43
--no-record-count-unit ⟨rule⟩ ............................................. 43
--retain-formats ⟨list⟩ ...................................................... 44
--no-retain-formats ........................................................ 44

3.5 Bib File Options .......................................................... 44
--warn-non-bib-fields ...................................................... 44
--no-warn-non-bib-fields .................................................. 44
--warn-unknown-entry-types .............................................. 45
--no-warn-unknown-entry-types ......................................... 45

3.6 Field Options ............................................................. 45
--no-expand-fields ......................................................... 45
--expand-fields .............................................................. 45
--mfirstuc-protection ⟨list⟩|all ⟨or -u ⟨list⟩|all⟩ ....................... 45
--no-mfirstuc-protection ................................................... 46
--mfirstuc-math-protection ............................................... 46
--no-mfirstuc-math-protection .......................................... 46
--nested-link-check ⟨list⟩|none ........................................... 46
--no-nested-link-check ..................................................... 46
--shortcuts ⟨value⟩ .......................................................... 47
--trim-fields ................................................................. 47
--trim-only-fields ⟨list⟩ ..................................................... 48
--trim-except-fields ⟨list⟩ ................................................. 48
--no-trim-fields ............................................................ 48
3.7 Other Options .............................................. 48
  --force-cross-resource-refs (or -x) .................. 48
  --no-force-cross-resource-refs ...................... 48
  --provide-glossaries ................................. 48
  --no-provide-glossaries ............................ 49
  --replace-quotes ..................................... 49
  --no-replace-quotes ................................ 49

4 .bib Format .................................................. 50
  4.1 Encoding ............................................... 50
  4.2 Comments ............................................... 51
  4.3 Fields .................................................. 51
  4.4 String Concatenation ................................ 59
  4.5 Standard Entry Types ............................... 59
    @string ............................................... 59
    @preamble ............................................. 60
  4.6 Single Entry Types ................................ 64
    @entry ................................................ 65
    @symbol .............................................. 66
    @number .............................................. 66
    @index ................................................. 67
    @indexplural ......................................... 68
    @abbreviation ....................................... 69
    @acronym .............................................. 70
    @contributor ......................................... 70
  4.7 Dual Entry Types .................................. 71
    @dualentry .......................................... 78
    @dualindexentry ..................................... 80
    @dualindexabbreviation ............................ 82
    @dualindexsymbol .................................. 83
    @dualindexnumber .................................. 87
    @dualabbreviationentry ............................ 87
    @dualentryabbreviation ............................ 88
    @dualsymbol .......................................... 89
    @dualnumber .......................................... 89
    @dualabbreviation .................................. 90
    @dualacronym ......................................... 95
  4.8 Tertiary Entry Types ............................... 95
    @tertiaryindexabbreviationentry ................. 96
  4.9 Multi-Entry Types ................................. 97
    @bibtexentry ......................................... 97
    @progenitor ......................................... 101
    @spawnindex .......................................... 104
    @spawnindexplural .................................. 104
5 Resource File Options

5.1 String Concatenation ............................................ 114
5.2 Complex Conditionals ........................................... 119
5.3 General Options .................................................. 124
 charset=⟨encoding-name⟩ ........................................ 124
 locale=⟨lang tag⟩ .................................................. 124
 interpret-preamble=⟨boolean⟩ .................................. 124
 write-preamble=⟨boolean⟩ ....................................... 124
 set-widest=⟨boolean⟩ ............................................ 124
 entry-type-aliases=⟨key=value list⟩ ...................... 126
 unknown-entry-alias=⟨value⟩ .................................. 127
 action=⟨value⟩ .................................................... 128
 copy-to-glossary=⟨list⟩ .......................................... 130
 copy-to-glossary-missing-field-action=⟨value⟩ ............ 131

5.4 Selection Options ............................................... 132
 src=⟨list⟩ ............................................................ 132
 selection=⟨value⟩ ................................................ 133
 match=⟨key=value list⟩ ........................................... 136
 match-op=⟨value⟩ ............................................... 137
 not-match=⟨key=value list⟩ ..................................... 137
 match-action=⟨value⟩ ............................................ 137
 limit=⟨number⟩ .................................................... 137

5.5 Hierarchical Options ............................................ 138
 save-child-count=⟨boolean⟩ .................................... 138
 save-sibling-count=⟨boolean⟩ ................................ 140
 save-root-ancestor=⟨boolean⟩ ................................ 140
 flatten=⟨boolean⟩ ................................................ 140
 flatten-lonely=⟨value⟩ ......................................... 140
 flatten-lonely-rule=⟨value⟩ ................................ 147
 flatten-lonely-condition=⟨value⟩ ...................... 148
 flatten-lonely-missing-field-action=⟨value⟩ ............ 148
 strip-missing-parents=⟨boolean⟩ ......................... 148
 missing-parents=⟨value⟩ ...................................... 149
 missing-parent-category=⟨value⟩ ......................... 150
 group-level=⟨value⟩ .............................................. 150
merge-small-groups=⟨n⟩ ................................................. 151

5.6 Master Documents .................................................. 152
master=⟨name⟩ ............................................................ 154
master-resources=⟨list⟩ ................................................ 156

5.7 Field and Label Options ............................................ 156
Label Options ............................................................. 156
interpret-label-fields=⟨boolean⟩ ..................................... 156
labelify=⟨list⟩ ............................................................ 157
labelify-list=⟨list⟩ ....................................................... 158
labelify-replace=⟨list⟩ .................................................. 159
label-prefix=⟨tag⟩ ......................................................... 160
duplicate-label-suffix=⟨value⟩ ....................................... 161
record-label-prefix=⟨tag⟩ .............................................. 162
cs-label-prefix=⟨tag⟩ ................................................... 162
ext-prefixes=⟨list⟩ ........................................................ 163
prefix-only-existing=⟨boolean⟩ ....................................... 165
save-original-id=⟨value⟩ ............................................... 165
save-original-id-action=⟨value⟩ ....................................... 165
save-definition-index=⟨boolean⟩ ..................................... 165
save-use-index=⟨boolean⟩ .............................................. 165
dependency-fields=⟨list⟩ ................................................. 166
Assignments ............................................................... 169
group=⟨label⟩ ............................................................. 169
category=⟨value⟩ .......................................................... 169
type=⟨value⟩ ............................................................... 171
trigger-type=⟨type⟩ ....................................................... 172
progenitor-type=⟨type⟩ .................................................. 172
progeny-type=⟨type⟩ ....................................................... 172
adopted-parent-field=⟨type⟩ ........................................... 172
ignore-fields=⟨list⟩ ........................................................ 173
field-aliases={⟨key=value list⟩} ....................................... 173
replicate-fields={⟨key=value list⟩} ................................... 174
replicate-override={⟨boolean⟩} ...................................... 176
replicate-missing-field-action={⟨value⟩} .......................... 176
assign-fields={⟨key=value list⟩} ..................................... 176
assign-override={⟨boolean⟩} .......................................... 181
assign-missing-field-action={⟨value⟩} .............................. 181
counter=⟨value⟩ .......................................................... 182
copy-action-group-field=⟨value⟩ ..................................... 182
copy-alias-to-see=⟨boolean⟩ .......................................... 182
save-from-see=⟨value⟩ .................................................. 183
save-from-seealso=⟨value⟩ .............................................. 183
save-from-alias=⟨value⟩ ................................................ 183
save-crossref-tail=⟨value⟩ .............................................. 183
Contents

description-case-change=(value) ........................................ 216
field-case-change={key=value list}) .............................. 216

5.8 Field Fallbacks .................................................. 216
abbreviation-name-fallback=(field) .................................. 219
abbreviation-text-fallback=(field) .................................. 219
text-sort-fallback=(field) ........................................... 219
entry-sort-fallback=(field) ........................................... 220
symbol-sort-fallback=(field) ........................................... 221
bibtexentry-sort-fallback=(field) ................................... 222
custom-sort-fallbacks={key=value list}) .......................... 222
field-concat-sep=(value) ............................................. 224

5.9 Plurals ............................................................. 225
short-plural-suffix=(value) ........................................... 227
dual-short-plural-suffix=(value) ..................................... 227

5.10 Location List Options ........................................... 227
save-locations=(value) ................................................ 231
save-loclist=(boolean) .................................................. 232
save-primary-locations=(value) ..................................... 232
primary-location-formats=(list) ...................................... 233
principal-location-formats=(list) ..................................... 234
primary-loc-counters=(value) ....................................... 238
principal-loc-counters=(value) ..................................... 238
merge-ranges=(boolean) ............................................... 240
min-loc-range=(value) .................................................. 241
max-loc-diff=(value) ................................................... 243
suffixF=(value) .......................................................... 244
suffixFF=(value) .......................................................... 244
compact-ranges=(value) ............................................... 244
see=(value) ............................................................... 245
seealso=(value) ........................................................... 245
alias=(value) ............................................................. 245
alias-loc=(value) .......................................................... 246
loc-prefix=(value) .......................................................... 246
loc-prefix-def=(value) ..................................................... 247
loc-suffix=(value) .......................................................... 248
loc-suffix-def=(value) ..................................................... 248
loc-counters=(list) .......................................................... 249
save-index-counter=(value) ............................................ 250

5.11 Supplemental Locations ......................................... 253
supplemental-locations=(basename) ................................ 255
supplemental-selection=(value) ....................................... 257
supplemental-category=(value) ....................................... 258
5.12 Sorting ................................................................. 258
sort=(value) .......................................................... 260
No Sort Field ......................................................... 261
Alphabet ............................................................. 265
Letter (Non Locale) ............................................... 267
Letter-Number .................................................. 268
Numerical ......................................................... 271
Date-Time .......................................................... 272
shuffle=(seed) ..................................................... 274
sort-field=(field) .................................................. 274
missing-sort-fallback=(field) ......................... 275
trim-sort=(boolean) ............................................. 276
sort-replace=(list) ................................................. 276
sort-rule=(value) .................................................. 277
break-at=(option) ............................................... 279
break-marker=(marker) ......................................... 281
break-at-match=(key=value list) ..................... 281
break-at-match-op=(value) ......................... 281
break-at-not-match=(key=value list) ............ 281
sort-number-pad=(number) .............................. 282
sort-pad-plus=(marker) ................................. 282
sort-pad-minus=(marker) ............................. 282
identical-sort-action=(value) ....................... 282
sort-suffix=(value) .............................................. 283
sort-suffix-marker=(value) ......................... 288
encapsulate-sort={csname} ......................... 288
strength=(value) .................................................. 288
decomposition=(value) ...................................... 289
letter-number-rule=(value) ......................... 289
letter-number-punc-rule=(value) .................. 290
numeric-sort-pattern=(value) ..................... 292
numeric-locale=(value) ................................... 292
date-sort-locale=(value) .............................. 292
date-sort-format=(value) ......................... 293
group-formation=(value) ........................... 295
5.13 Secondary Glossary ........................................ 295
secondary=(value) ............................................. 295
secondary-match=(key=value list) ............... 298
secondary-not-match=(key=value list) ........ 298
secondary-match-op=(value) ....................... 298
secondary-match-action=(value) ............... 298
secondary-missing-sort-fallback=(field) ...... 298
secondary-trim-sort=(boolean) .................. 299
secondary-sort-replace=(list) ................. 299
secondary-sort-rule={value} ................................................. 299
secondary-break-at={value} .................................................. 299
secondary-break-marker={marker} .......................................... 299
  secondary-break-at-match={key=value list} ............................... 299
  secondary-break-at-match-op={value} ................................... 299
  secondary-break-at-not-match={key=value list} ......................... 299
secondary-sort-number-pad={number} ....................................... 299
secondary-sort-pad-plus={marker} ......................................... 299
secondary-sort-pad-minus={marker} ....................................... 299
secondary-identical-sort-action={value} .................................. 300
secondary-sort-suffix={value} ............................................... 300
secondary-sort-suffix-marker={value} ................................... 300
secondary-strength={value} .................................................. 300
secondary-decomposition={value} .......................................... 300
secondary-letter-number-rule={value} .................................... 300
secondary-letter-number-punc-rule={value} ............................. 300
secondary-numeric-sort-pattern={value} .................................. 300
secondary-numeric-locale={value} .......................................... 300
secondary-date-sort-locale={value} ........................................ 300
primary-dual-dependency={boolean} ...................................... 301
combine-dual-locations={value} ............................................. 301
dual-type={value} .............................................................. 303
dual-category={value} ......................................................... 304
dual-counter={value} .......................................................... 305
dual-short-case-change={value} ............................................. 305
dual-long-case-change={value} .............................................. 305
dual-field={value} .............................................................. 305
dual-date-time-field-format={value} ...................................... 306
dual-date-field-format={value} .............................................. 306
dual-time-field-format={value} ............................................. 306
dual-date-time-field-locale={value} ...................................... 306
dual-date-field-locale={value} ............................................. 306
date-time-field-locale={value} ............................................ 306
5.14 Dual Entries ................................................................. 301
  General Dual Settings .......................................................... 301
dual-prefix={value} ............................................................. 301
  primary-dual-dependency={boolean} ...................................... 301
combine-dual-locations={value} ............................................. 301
  Dual Fields ........................................................................ 303
dual-type={value} .............................................................. 303
dual-category={value} ......................................................... 304
dual-counter={value} .......................................................... 305
dual-short-case-change={value} ............................................. 305
dual-long-case-change={value} .............................................. 305
dual-field={value} .............................................................. 305
dual-date-time-field-format={value} ...................................... 306
dual-date-field-format={value} .............................................. 306
dual-time-field-format={value} ............................................. 306
dual-date-time-field-locale={value} ...................................... 306
dual-date-field-locale={value} ............................................. 306
date-time-field-locale={value} ............................................ 306
  Dual Sorting ........................................................................ 306
dual-sort={value} .............................................................. 306
dual-sort-field={field} ......................................................... 307
dual-missing-sort-fallback={field} .......................................... 307
dual-trim-sort={boolean} ....................................................... 307
dual-sort-replace=(list) ........................................... 307
dual-sort-rule=(value) .............................................. 307
dual-break-at=(value) ................................................. 307
dual-break-marker=(marker) ......................................... 308
dual-break-at-match=(key=value list) ............................ 308
dual-break-at-match-op=(value) .................................... 308
dual-break-at-not-match=(key=value list) ......................... 308
dual-sort-number-pad=(number) .................................... 308
dual-sort-pad-plus=(marker) ......................................... 308
dual-sort-pad-minus=(marker) ....................................... 308
dual-identical-sort-action=(value) ................................. 308
dual-sort-suffix=(value) .............................................. 308
dual-sort-suffix-marker=(value) .................................... 308
dual-strength=(value) ................................................ 308
dual-decomposition=(value) .......................................... 308
dual-letter-number-rule=(value) .................................... 309
dual-letter-number-punc-rule=(value) ............................. 309
dual-numeric-sort-pattern=(value) ................................ 309
dual-numeric-locale=(value) ........................................ 309
dual-date-sort-locale=(value) ....................................... 309
dual-date-sort-format=(value) ....................................... 309
dual-group-formation=(value) ....................................... 309

Dual Mappings ......................................................... 309
dual-entry-map={{(list1)},{(list2)}} .............................. 309
dual-abbrv-map={{(list1)},{(list2)}} ............................... 310
dual-abbrventry-map={{(list1)},{(list2)}} ....................... 311
dual-symbol-map={{(list1)},{(list2)}} ............................ 311
dual-indexentry-map={{(list1)},{(list2)}} ....................... 311
dual-indexsymbol-map={{(list1)},{(list2)}} ..................... 311
dual-indexabbrv-map={{(list1)},{(list2)}} ....................... 312

Dual Back-Links ...................................................... 312
dual-entry-backlink={{boolean}} .................................... 312
dual-abbrv-backlink={{boolean}} ................................... 313
dual-symbol-backlink={{boolean}} ................................ 313
dual-abbrventry-backlink={{boolean}} ............................ 313
dual-indexentry-backlink={{boolean}} ............................ 314
dual-indexsymbol-backlink={{boolean}} .......................... 314
dual-indexabbrv-backlink={{boolean}} ........................... 314
dual-backlink={{boolean}} ............................................ 314

5.15 Tertiary Entries ................................................. 314
tertiary-prefix={{(value)}} ........................................... 314
tertiary-type={{(value)}} ............................................. 314
tertiary-category={{(value)}} ....................................... 315
## 6 Provided Commands

### 6.1 Entry Definitions

- \bibglsnewentry .................................................. 319
- \bibglsnewsymbol .................................................. 320
- \bibglsnewnumber .................................................. 320
- \bibglsnewindex ...................................................... 321
- \bibglsnewindexplural ................................................ 321
- \bibglsnewababbreviation ............................................. 321
- \bibglsnewacronym .................................................... 322
- \bibglsnewdualentry .................................................. 322
- \bibglsnewdualindexentry ............................................. 322
- \bibglsnewdualindexentrysecondary ............................... 322
- \bibglsnewdualindexsymbol .......................................... 323
- \bibglsnewdualindexsymbolsecondary ......................... 323
- \bibglsnewdualindexnumber ............................................ 323
- \bibglsnewdualindexnumbersecondary ....................... 323
- \bibglsnewdualindexabbreviation .................................. 324
- \bibglsnewdualindexabbreviationsecondary .................... 324
- \bibglsnewdualababbreviationentry ............................. 325
- \bibglsnewdualababbreviationentrysecondary .............. 325
- \bibglsnewdualentryabbreviation .................................. 325
- \bibglsnewdualentryabbreviationsecondary ............... 326
- \bibglsnewdualsymbol ................................................. 326
- \bibglsnewdualnumber .................................................. 326
- \bibglsnewdualacronym ............................................... 327
- \bibglsnewtertiaryindexabbreviationentry ....................... 327
- \bibglsnewtertiaryindexabbreviationentrysecondary ........ 328
- \bibglsnewbitextentry .................................................. 328
- \bibglsnewcontributor .................................................. 328
- \bibglsnewprogenitor .................................................... 329
- \bibglsnewspawnindex .................................................. 329
- \bibglsnewspawnedindex .................................................. 329
Contents

\bibglssetemptygrouptitlehier .................................. 356
\bibglseemptygrouphier ............................................. 356
\bibglseemptygrouptitlehier ......................................... 357
\bibglssetnumbergrouptitlehier ...................................... 357
\bibglsnnumbergrouphier ............................................. 357
\bibglssetdatetimegrouptitlehier .................................... 357
\bibglsdatetimegrouphier ........................................... 358
\bibglsdatetimegrouphierfinalargs .................................. 358
\bibglssetdategrouptitlehier ......................................... 359
\bibglstypegrouphier .................................................. 359
\bibglssettimegrouptitlehier ......................................... 359
\bibglstimegrouphier .................................................. 359
\bibglstimegrouptitlehier ............................................ 360
\bibglssetunicodegrouptitlehier .................................... 360
\bibglsunicodegrouphier .............................................. 360
\bibglsunicodegrouptitlehier ......................................... 360
\bibglssetmergedgrouptitlehier ....................................... 360
\bibglsmergedgrouphier .............................................. 361
\bibglsmergedgrouptitlehier ......................................... 361
\bibglsmergedgrouphierfmt ............................................ 361

6.5 Flattened Entries .................................................. 362
\bibglssflattenedhomograph .......................................... 362
\bibglssflattenedchildpresort ........................................ 363
\bibglssflattenedchildpostsort ....................................... 363

6.6 Other ........................................................................ 364
\bibglscopytoglossary ................................................... 364
\bibglsssettotallrecordcount ......................................... 364
\bibglssetrecordcount ................................................. 364
\bibglssetlocationrecordcount ........................................ 365
\bibglsshypervisor ..................................................... 365
\bibglssetwidest .......................................................... 365
\bibglssetwidestfortype ................................................. 366
\bibglssetwidestfallback ................................................ 366
\bibglssetwidestfortypefallback ...................................... 366
\bibglssetwidesttoplevelfallback ..................................... 367
\bibglssetwidesttoplevelfortypefallback ............................. 367
\bibglsscontributorlist .................................................. 367
\bibglsscontributor ..................................................... 368
\bibglssdatetime .......................................................... 368
\bibglssdate ................................................................ 368
7 Converting Existing .tex to .bib 372
7.1 Command Line Arguments ............................................................ 372
--texenc ⟨encoding⟩ ........................................................................ 372
--bibenc ⟨encoding⟩ ........................................................................ 372
--space-sub ⟨replacement⟩ (or -s ⟨replacement⟩) ................................ 372
--ignore-sort .................................................................................. 373
--no-ignore-sort .............................................................................. 373
--ignore-type ................................................................................... 373
--no-ignore-type ............................................................................. 373
--split-on-type (or -t) ................................................................. 373
--no-split-on-type ...................................................................... 373
--ignore-category ........................................................................ 374
--no-ignore-category ................................................................... 374
--split-on-category (or -c) ............................................................. 374
--no-split-on-category ................................................................. 374
--ignore-fields ⟨list⟩ (or -f ⟨list⟩) .................................................. 374
--preamble-only (or -p) ................................................................. 375
--no-preamble-only ................................................................. 375
--absorb-see ................................................................................. 375
--no-absorb-see .......................................................................... 375
--index-conversion (or -i) ............................................................. 375
--no-index-conversion ................................................................. 375
--locale ⟨language tag⟩ ................................................................. 375
--overwrite .................................................................................... 376
--no-overwrite .............................................................................. 376
--silent .................................................. 376
--verbose ............................................... 376
--debug ................................................ 376
--help (or -h) ......................................... 376
--version (or -v) ...................................... 376

7.2 Recognised Commands .................................. 376
  \gsexpandfields ....................................... 377
  \glsnoexpandfields ................................... 377
  \glsetexpandfield ..................................... 377
  \glsetnoexpandfield ................................... 378
  \newglossaryentry .................................... 378
  \provideglossaryentry ................................ 378
  \longnewglossaryentry ................................ 378
  \longprovideglossaryentry ......................... 379
  \newterm .............................................. 379
  \newabbreviation .................................... 380
  \newacronym .......................................... 380
  \glsxtrnewsymbol ..................................... 381
  \glsxtrnewnumber ..................................... 381
  \newdualentry ........................................ 382

8 Examples ............................................. 384
  no-interpret-preamble.bib .............................. 384
  interpret-preamble.bib ............................... 385
  interpret-preamble2.bib .............................. 385
  constants.bib ......................................... 386
  chemicalformula.bib .................................. 389
  bacteria.bib .......................................... 393
  baseunits.bib ......................................... 395
  derivedunits.bib ...................................... 397
  people.bib ............................................ 398
  books.bib ............................................. 404
  films.bib ............................................. 407
  citations.bib ......................................... 413
  mathgreek.bib ........................................ 414
  bigmathsymbols.bib .................................... 419
  mathsrelations.bib .................................... 423
  binaryoperators.bib ................................... 425
  unaryoperators.bib .................................... 426
  mathobjects.bib ....................................... 427
  miscsymbols.bib ....................................... 431
  markuplanguages.bib .................................. 435
  usergroups.bib ....................................... 437
  animals.bib .......................................... 442
Contents

minerals.bib ........................................ 444
vegetables.bib ........................................ 446
terms.bib ........................................... 448
topics.bib ........................................... 449
sample-hierarchical.tex ......................... 449
sample-nested.tex ................................ 450
sample-constants.tex ...................... 455
sample-chemical.tex ......................... 460
sample-bacteria.tex .......................... 463
sample-units1.tex ............................. 467
sample-units2.tex ............................. 470
sample-units3.tex ............................. 473
sample-media.tex ............................. 478
sample-people.tex ............................. 482
sample-authors.tex ........................... 490
sample-citations.tex ......................... 494
sample-msymbols.tex ......................... 499
sample-maths.tex ............................. 501
sample-textsymbols.tex ..................... 506
sample-textsymbols2.tex .................... 509
sample-markuplanguages.tex ................... 512
sample-usergroups.tex ...................... 516
sample-multi1.tex ............................. 524
sample-multi2.tex ............................. 535

Package Option Summary .......................... 561
General Command Summary ........................ 567
Bibliography ........................................ 641
Index .............................................. 643
List of Tables

2.1 Glossary-Related Commands Implemented by the bib2gls Interpreter 23

4.1 Fields Provided by glossaries-extra 53
4.2 Fields Provided by bib2gls 54
4.3 Fields Provided by glossaries-prefix 54
4.4 Fields Provided by glossaries-accsupp 54
4.5 Fields Set by bib2gls 55
4.6 Internal Fields Set by glossaries or glossaries-extra or bib2gls 57
4.7 Compound Set Fields 57

5.1 Summary of Available Sort Options: No Sort Field 262
5.2 Summary of Available Sort Options: Alphabet 262
5.3 Summary of Available Sort Options: Letter (Non-Locale) 262
5.4 Summary of Available Sort Options: Letter-Number 262
5.5 Summary of Available Sort Options: Numerical 263
5.6 Summary of Available Sort Options: Date-Time 263
# List of Figures

5.1 Regular letter comparison vs letter-number comparison .......................... 269

8.1 sample-hierarchical.pdf ......................................................... 451
8.2 sample-nested.pdf ............................................................... 456
8.3 sample-constants.pdf .......................................................... 461
8.4 sample-chemical.pdf ........................................................... 464
8.5 sample-bacteria.pdf ............................................................ 468
8.6 sample-units1.pdf ............................................................... 471
8.7 sample-units2.pdf ............................................................... 474
8.8 sample-units3.pdf ............................................................... 478
8.9 sample-media.pdf ............................................................... 483
8.10 sample-people.pdf .............................................................. 491
8.11 sample-authors.pdf ............................................................. 495
8.12 sample-citations.pdf ........................................................... 499
8.13 sample-msymbols.pdf .......................................................... 502
8.14 sample-maths.pdf ............................................................... 507
8.15 sample-textsymbols.pdf ......................................................... 510
8.16 sample-textsymbols2.pdf ....................................................... 513
8.17 sample-markuplanguages.pdf ................................................. 517
8.18 sample-usergroups.pdf ......................................................... 525
8.19 sample-multi1.pdf (pages 1 to 4) ............................................ 536
8.20 sample-multi1.pdf (pages 5 to 8) ............................................ 537
8.21 sample-multi2.pdf (pages 1 to 4) ............................................ 558
8.22 sample-multi2.pdf (pages 5 to 8) ............................................ 559
8.23 sample-multi2.pdf (pages 9 and 12) ........................................ 560
Glossary

Ancestor
An entry’s parent or an ancestor of the parent. See section 5.5.

Anchored (Regular Expression)
An anchored regular expression must match the entire string, not a sub-string. For example, `l?op` matches “lop” and “op” but doesn’t match “clop” or “cop”.

Child Entry
An entry in a hierarchical glossary that is linked to, but one level down from, its associated parent entry. See section 5.5.

Compound (Combined or Multi) Entry
A compound entry corresponds to the \multiglossaryentry command. This defines a label that represents a set of entries that have already been defined. This label can then be used in commands like \gls as a shortcut for using \gls for each element in the set. The main label is the main element in the set. The “other labels” are all the other (not-main) elements. See section 4.10.

Concatenation
This is where multiple fragments or substrings are joined together to form a single value. The concatenation operator is `#` for .bib files (see section 4.4) and `+` for resource option string concatenation (section 5.1).

Cross-reference Field
A field used for cross-referencing another entry: see, seealso and alias. Other fields can be identified as a list of dependent entry labels with dependency-fields.

Cross-resource Reference
A reference from a recorded entry provided in one resource set to an unrecorded entry in another resource set. See section 1.4.

Definition Index
An index (starting from 0) that’s incremented every time a new entry object is created within bib2gls. This relates to the order of definitions within the .bib files. Each dual entry and spawned entry will increment the underlying counter but only when they are created, which may not happen until after all .bib files for the resource set have been parsed.
Discarded Record

A record that is discarded because either it is identical to another record or it conflicts with another record.

Document Locale

the locale associated with the document language (or by --locale, if no document language has been detected). In the case of a multi-lingual document, this is the locale of the last language resource file to be loaded through tracklang’s interface. It’s best to explicitly set the locale for multi-lingual documents to avoid confusion (either with the locale or as a language tag in options such as sort).

Dual Entry

The duplicate entry created from a dual-entry type (such as @dualentry). This duplicate is based on the primary entry with modifications made according to various settings. With tertiary entry types, the dual entry represents two entries: the secondary and tertiary. See section 4.7.

Dual List

The bib2glsls list of dual entries, which is sorted according to the dual-sort resource option. The entries may or may not be assigned to the same glossary, and the list may only be a subset of entries. If dual-sort={combine} is used then all entries will be in the main list and there won’t be a dual list.

Encoding

A text format that maps a byte or sequence of bytes to a character. See section 4.1 and --tex-encoding for the .bib file encoding and charset for the .glstex file encoding. See also the blog article Binary Files, Text Files and File Encodings for further information about file encodings in general.

Entry Type

an entry’s identifying type, as specified by @entry-type. (Not to be confused with the glossary label, which is identified by the type field.) When referenced in a resource option, the leading @ is typically omitted. The original entry type refers to the entry type as specified in the .bib file. The actual entry type may be different and will be the result of a conversion via resource options such as entry-type-aliases. Although the .bib format is case-insensitive, references to the entry type in resource options should typically be in lower case.

Flat Glossary

A glossary that has no hierarchy. That is, there are no child entries. See section 5.5.
Hierarchical Glossary

A glossary where the entries are ranked according to some classification. Level 0 indicates top-level entries, level 1 indicates child entries that have a level 0 parent, level 2 indicates child entries that have a level 1 parent, and so on. See section 5.5.

Homograph

Each word in a set of words that all have the same spelling but different meanings. For example, lead (to guide someone) and lead (metallic element) are homographs.

Identical Collator Strength

A collator strength value that indicates that all differences are considered significant during comparison.

Ignored Glossary

A glossary defined with commands like \newignoredglossary. An ignored glossary doesn’t have an associated title (so if one is required it needs to be explicitly set), and isn’t picked up by iterative commands such as \printunsrtglossaries. See section 1.3.

Ignored Record

A record with the format glsignore or glstriggerrecordformat. This record indicates that the entry should be considered for selection with any of the “recorded” selection options, but the record should not be added to the location list.

Java Locale

the default locale for the Java Runtime Environment (JRE), which usually matches the operating system’s locale.

Location

The value of the indexing counter when an entry is recorded. By default, this is the page counter. Each location has an associated format or encapsulating command (ENCAP), which is the name of a formatting command that should be used to encapsulate the location’s value in the location list. The default is glsnumberformat.

Location List

Formatted list of locations obtained from an entry’s records. This won’t include ignored or discarded records, and a run of locations may be compressed into a range. See section 5.10 and section 6.3.

Lonely Child Entry

A child entry that has no selected siblings. See section 5.5.
Main Document

The principal document that has its own glossary but the location lists may also contain external locations obtained from a supplemental document.

Main Entry

The originating entry from which the spawned entries are created. A main entry may be a dual-entry type, consisting of a primary entry and dual entry. (Not to be confused with the main glossary or the main label of a compound entry.)

Main Glossary

The default glossary in the document identified by \glsdefaulttype (which will have the label main unless nomain is used). If nomain is used then \glsdefaulttype will be set to the label of the first glossary to be defined.

Main Label or Element (Compound Entry)

The main element in the set that defines a compound entry.

Main (or Primary) List

The \bib2gls list of primary entries, which is sorted according to the sort resource option. The entries may or may not be assigned to the same glossary, and the list may only be a subset of entries. If dual-sort={combine} is used, then the main list will also contain all the dual entries.

Master Document

A main or principal document that contains a glossary with entries referenced by smaller documents that don’t have their own glossary. See section 5.6.

Multi-entry Type

An entry type that can spawn multiple primary entries. Some multi-entry types can also spawn a dual entry. See section 4.9. For the glossaries-extra “multi (compound or combined) entries” that are defined with \multiglossaryentry see compound entry.

Order of Use Index

The record index is a value (starting from 0) that’s incremented every time a record is created while parsing the .aux file. The first time a non-ignored record is added to a given entry, the record index is assigned to that entry’s order of use index. So the index provides a relative order of use. So if entry1 is the first entry to be indexed, it will have order of use index 0. If entry1 is then indexed twice more and then entry2 is indexed, then entry2’s order of use index will be 3.

Other Label or Element (Compound Entry)

The non-main elements in the set that defines a compound entry.
**Parent Entry**

An entry in a hierarchical glossary that is linked to, but one level up from, its associated child entry. See section 5.5.

**Primary Collator Strength**

A collator strength value that indicates only primary differences are considered significant during comparison. This is locale dependant, but typically different base letters are considered a primary difference.

**Primary Entry**

The original entry created from a dual-entry type (such as @dualentry) or the entry from single-entry types (such as @entry) or spawned entries.

**Primary (or Principal) Glossary**

A glossary that contains entries that have the type field set to that glossary’s label. Note that a primary glossary may contain both primary and dual entries.

**Principal (or Primary) Location**

A special location (record) which indicates the principal or primary place in the document where the entry is mentioned or discussed. The location is identified by the principal or primary format (principal-location-formats).

**Progenitor**

The main entry for the @progenitor entry type.

**Progeny**

The spawned entries for the @progenitor entry type.

**Record**

Recording is bib2gls’s equivalent of indexing. When the record package option is set, each time an entry is indexed in the document (using commands like \gls or \glstext) a record is added to the .aux file that makes a note of the entry label, the location, the counter that was used to obtain the location, and (optionally) hyperlink information. A record may be ignored or discarded but, regardless of this, if an entry has at least one record it will be considered for selection for any of the “recorded” selection options.

**Record Count**

An entry’s record count is the total number of records (including discarded and ignored) written to the .aux file that are associated with the entry. It’s also possible to have sub-totals for each record counter.

**Recorded Entry**

An entry that has one or more records.
Glossary

Regular Expression
A pattern that specifies how to match text. Unless indicated otherwise, resource options that use regular expressions are anchored. See Java’s Pattern class API [5] for details of the regular expression syntax.

Resource Command
\glslxtrresourcefile or \GlsXtrLoadResources.

Resource Locale
the default locale for the given resource set. This can be set with the locale resource option. If not explicitly set, then the default will be the document language, if it has been detected by tracklang or identified with --locale, or the JRE locale otherwise.

Resource Set
The set of options and entries associated with a resource command. See section 1.4.

Secondary Collator Strength
A collator strength value that indicates only primary and secondary differences are considered significant during comparison. This is locale dependant. For example, in some languages different accented forms of the same base letter may be considered a secondary difference.

Secondary Entry
For the tertiary entry types, such as @tertiaryindexabbreviationentry, there are only actually two objects defined within bib2gls: the primary and the dual, but the code that is written in the .glstex file for the dual entry actually defines two entries, which are the secondary and tertiary entries. This should not be confused with the secondary glossary. See section 4.8.

Secondary Glossary
A secondary glossary is one that contains labels of entries that have been defined for another glossary. The actual entry’s type field will be set to the primary glossary.

Sibling Entry
Two or more child entries are siblings if they all share the same parent entry. See section 5.5.

Spawned Entry
A duplicate entry created from a multi-entry type (such as @spawnentry).

Sub-entry
A child entry. More specifically, when contrasted with sub-sub-entry etc, this may refer to level 1 entries (which have a parent that is a top-level entry). See section 5.5.
Glossary

**Supplemental (or Supplementary) Document**
A related document from which supplemental records are obtained.

**Supplemental Record**
A record obtained from another document. See section 5.11.

**Tertiary Collator Strength**
A collator strength value that indicates only primary, secondary and tertiary differences are considered significant during comparison. This is locale dependant. For example, different cases of the same base letter may be considered a tertiary difference.

**Tertiary Entry**
An entry that isn’t defined as a separated object within bib2gls, but is defined within the .glstex file as a by-product of the dual definition code for tertiary entry types.

**Top-level Entry**
An entry that doesn’t have a parent entry. This entry is the hierarchical root for all its descendents. See section 5.5.

**Unrecorded Entry**
An entry that doesn’t have any records.
1 Introduction

If you have extensively used the glossaries [14] or glossaries-extra [13] package, you may have found yourself creating a large .tex file containing many definitions that you frequently use in documents. This file can then simply be loaded using \input or \loadglsentries, but a large file like this can be difficult to maintain and if the document only actually uses a small proportion of those entries, the document build is unnecessarily slow due to the time and resources taken on defining the unwanted entries.

The aim of bib2gls is to allow the entries to be stored in a .bib file, which can be maintained using a reference system such as JabRef. The document build process can now be analogous to that used with bibtex (or biber), where only those entries that have been recorded in the document (and possibly their dependent entries) will be extracted from the .bib file. Since bib2gls can also perform hierarchical sorting and can collate location lists, it doubles as an indexing application, which means that the makeglossaries step can be skipped. Note that bib2gls doesn’t warn you if an entry that’s referenced in the document doesn’t exist in any of the supplied .bib files, but instead relies on the glossaries-extra package to generate the warning. So at the end of the document build check the .log file for warnings.

You can’t use \glsaddall with bib2gls as that command works by iterating over all defined entries and calling \glsadd{⟨label⟩}. On the first LaTeX run there are no entries defined, so \glsaddall does nothing. If you want to select all entries, just use selection={all} instead (which has the advantage over \glsaddall in that it doesn’t create a redundant location for each entry).

Note that bib2gls requires the extension package glossaries-extra and can’t be used with just the base glossaries package, since it requires some of the extension commands. See the glossaries-extra user manual [13] for information on the differences between the basic package and the extended package, as some of the default settings are different.

Since the information used by bib2gls is written to the .aux file, it’s not possible to run bib2gls through LaTeX’s shell escape while the .aux file is open for write access. (The .aux file is closed after the end document hook, so it can’t be deferred with \AtEndDocument.) This means that if you really want to run bib2gls through \write18 it must be done in the preamble with \immediate. For example:

\immediate\write18{bib2gls \jobname}

As from version 1.14 of glossaries-extra, this can be done automatically with the automake option if the .aux file exists. (Remember that this will require the shell escape to be enabled.)
1.1 Example Use

The glossary entries are stored in a .bib file. For example, the file entries.bib might contain:

@entry{bird,
   name={bird},
   description = {feathered animal}
}

@abbreviation{html,
   short="html",
   long={hypertext markup language}
}

@symbol{v,
   name=${\vec{v}}$,
   text={\vec{v}},
   description={a vector}
}

@index{goose,plural="geese"}

Here's an example document that uses this data:

\documentclass{article}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[
   src={entries},% data in entries.bib
   sort={en-GB}% sort according to 'en-GB' locale
]
\begin{document}
\Gls{bird} and \gls{goose}. Symbol: $\gls{v}$. Abbreviation: \gls{html}.
\printunsrtglossaries
\end{document}

If this document is called myDoc.tex, the build process is:

pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
1.1 Example Use

(This manual assumes pdflatex for simplicity. Replace with latex, xelatex or lualatex as appropriate.) If you want letter groups (either headed, with styles like indexgroup, or just a blank line separator with nogroupskip={false}) then you need to use the --group switch:

```
pdflatex myDoc
bib2gls --group myDoc
pdflatex myDoc
```

You can have multiple instances of \GlsXtrLoadResources. For example:

```
\documentclass{article}
\usepackage[record,index,abbreviations,symbols]{glossaries-extra}
\GlsXtrLoadResources[
    src={entries},% data in entries.bib
    sort={en-GB},% sort according to 'en-GB' locale
    match={entrytype={entry}},% only select @entry
    type={main}% put these entries in the 'main' glossary
]
\GlsXtrLoadResources[
    src={entries},% data in entries.bib
    sort={en-GB},% sort according to 'en-GB' locale
    match={entrytype={abbreviation}},% only select @abbreviation
    type={abbreviations}% put these in the 'abbreviations' glossary
]
\GlsXtrLoadResources[
    src={entries},% data in entries.bib
    sort={letter-case},% case-sensitive letter sort
    match={entrytype={symbol}},% only select @symbol
    type={symbols}% put these entries in the 'symbols' glossary
]
\GlsXtrLoadResources[
    src={entries},% data in entries.bib
    sort={en-GB},% sort according to 'en-GB' locale
    match={entrytype={index}},% only select @index
    type={index}% put these entries in the 'index' glossary
]
\begin{document}
\Gls{bird} and \gls{goose}.
Symbol: \textdollar\gls{v}\textdollar. Abbreviation: \gls{html}.
```
1.2 Logical Divisions: type vs group vs parent

\printunsrtglossaries
\end{document}

There are more examples provided in chapter 8.

Note that there’s no need to called xindy or makeindex since bib2gls automatically sorts the entries and collates the locations after selecting the required entries from the .bib file and before writing the temporary file that’s input with \glsxtrresourcefile (or the more convenient shortcut \GlsXtrLoadResources). This means the entries are already defined in the correct order, and only those entries that are required in the document are defined, so \printunsrtglossary (or \printunsrtglossaries) may be used. (The “unsrt” part of the command name indicates that all defined entries should be listed in the order of definition from glosaries-extra’s point of view, see the supplementary document “glosaries-extra and bib2gls: An Introductory Guide” (bib2gls-begin.pdf) for further details.)

If you don’t provide a value with the record option, then record={only} is assumed. This saves the same indexing information that’s used with the \makeglossaries and \makeindexglossaries methods (described in the main glosaries user manual [14]). As from glosaries-extra version 1.37, you can instead use record={nameref}, which saves some extra information for each location that’s not available for the other indexing methods. See \--merge-nameref-on for further details.

If you additionally want to use an indexing application, such as xindy, you need the package option record={alsoindex} and use \makeglossaries and \printglossary (or the iterative \printglossaries) as usual. This requires a more complicated build process:

\texttt{pdflatex myDoc}
\texttt{bib2gls myDoc}
\texttt{pdflatex myDoc}
\texttt{makeglossaries myDoc}
\texttt{pdflatex myDoc}

(The entries aren’t defined until the second \LaTeX run, so the indexing files required by xindy or makeindex can’t be created until then.) In this case, bib2gls is simply being used to fetch the entry definitions from one or more .bib files, with the sorting and collating performed by the other indexing application (so the resource option list would need sort={none} and save-locations={false}). In general, it’s best to avoid this hybrid method unless you have a particular set of xindy rules that can’t be replicated with bib2gls.

1.2 Logical Divisions: type vs group vs parent

If you have a document with many terms that need listing, it’s likely that you may want to divide the terms into separate blocks or units for easier reading. There are three fields that are used for this.

\footnote{This document will mostly use the more convenient \GlsXtrLoadResources.}
1.2 Logical Divisions: type vs group vs parent

type  The highest division is the glossary to which the entry belongs. The glossary must first be defined (see section 1.3) with an associated label used to identify it. The title is assigned to the glossary when it is defined or it can be overridden with the title key. The glossary is displayed using \printunsrtglossary and the title is placed in a sectioning command by default.

bib2gls does not provide any means of sorting glossary types. If you use \printunsrtglossaries the order will be according to the order in which the glossaries were defined. You may use \printunsrtglossary to list individual glossaries in your own preferred order.

group  The entries within a glossary can form groups as a by-product of the sorting method. This must be enabled with the --group switch and isn’t available for the sort methods listed in table 5.1. The group label is stored in the group field. This is an internal field that typically shouldn’t be set in the .bib file.

You can specify your own custom groups but if you do so you must ensure that the terms are ordered in such a way that they are gathered according to group. This is typically done by splitting the glossary into blocks using a separate \GlsXtrLoadResources with the group option set. You control the order of the groups by your ordering of \GlsXtrLoadResources. The group title can be assigned using \glsxtrsetgrouptitle within the document.

bib2gls does not sort by group title. At most it can sort by the group label (by changing the sort-field) but this is usually an indication that you actually have a hierarchical glossary and you ought to be using the parent field instead. (Compare sample-textsymbols.tex and sample-textsymbols2.tex.)

parent  An entry may have one or more sub-entries. Most of the sort methods will produce a hierarchical ordering that ensures that the sub-entries are listed immediately after their parent entry. The parent entry is identified by the parent field which should contain the parent’s label.

bib2gls sorts the parent and child entries using the same comparator. The sort methods listed in table 5.1 disregard the hierarchical level, which can result in child entries becoming detached from their parent entry. The other methods sort hierarchically using the same comparator but take the hierarchical level into account.

Suppose you have a mixture of terms, abbreviations and symbols, then you might want to have three glossaries that are listed in the table of contents. In this case, you use the type field or the type resource option. The ordering of the glossaries is determined by the ordering of the \printunsrtglossary commands within the document. For example:
1.2 Logical Divisions: type vs group vs parent

\printunsrtglossary
\printunsrtglossary[type={abbreviations}]
\printunsrtglossary[type={symbols}]

Suppose that your list of terms spans many pages and you feel it would be helpful to the reader to split it up into letter groups then you would need to run \texttt{bib2gls} with the \texttt{--group} switch and use a glossary style that supports letter groups for that glossary. For example:

\printunsrtglossary[style={indexgroup}]

Suppose that your list of symbols consists of pictographs, Latin characters and Greek characters and you want them grouped together in that order. Then you would use a separate \texttt{\GlsXtrLoadResources} for each block and assign your own custom group. This means ensuring that each resource set only selects the terms for that group. The simplest way of doing this is to have a separate \texttt{.bib} file for each set. For example:

\begin{verbatim}
\glsxtrsetgrouptitle{pictographs}{Pictographs}
\glsxtrsetgrouptitle{latinsymbols}{Latin Characters}
\glsxtrsetgrouptitle{greeksymbols}{Greek Characters}
\GlsXtrLoadResources[
  src={generalsymbols},% data in generalsymbols.bib
  group={pictographs},
  type={symbols}
]
\GlsXtrLoadResources[
  src={latinsymbols},% data in latinsymbols.bib
  group={latin},
  type={symbols}
]
\GlsXtrLoadResources[
  src={greeksymbols},% data in greeksymbols.bib
  group={greek},
  type={symbols}
]
\end{verbatim}

Suppose instead that you have many of these logical blocks and you want them ordered according to the block title. In this case you have a hierarchical glossary and you need to use the \texttt{parent} field. You then need to select an appropriate glossary style.

If you only want to have a single \texttt{.bib} file that contains all your entries and you want to share it across multiple documents then the most flexible approach is to use custom fields and entry types that can be aliased according to the needs of the resource sets.

For example, the file \texttt{entries.bib}:

\begin{verbatim}
% Encoding: UTF-8

@indexplural{latin,text={Latin character}}
\end{verbatim}
1.2 Logical Divisions: type vs group vs parent

@indexplural{greek, text={Greek character}}
@indexplural{pictograph}

@symbol{fx, name={\ensuremath{f(x)}}, description={function of $x$}, identifier={latin}}

@symbol{f'x, name={\ensuremath{f'(x)}}, description={derivative of \gls{fx}}, identifier={latin}}

@symbol{pi, name={\ensuremath{\pi}}, description={ratio of circumference to diameter}, identifier={greek}}

@symbol{heart, name={\ensuremath{\heartsuit}}, description={heart}, identifier={pictograph}}

@symbol{diamond, name={\ensuremath{\diamondsuit}}, description={diamond}, identifier={pictograph}}

@abbreviation{html, short={html}, long={hypertext markup language}, identifier={markuplanguage}}

@abbreviation{xml, short={xml}, long={extensible markup language}, identifier={markuplanguage}}
1.2 Logical Divisions: type vs group vs parent

This has a custom field identifier. This will be ignored by bib2gls unless defined or aliased in the document.

Here’s an example document that creates three glossary types (the default main glossary and the glossaries created with the abbreviations and symbols options). They are listed in the order of \printunsrtglossary and their titles are added to the table of contents.

The custom identifier fields are ignored for the main and abbreviation glossaries, but they are aliased for the symbols to the group field. Since I’ve split the symbols glossary into blocks with each block only containing entries that have the same group value, this isn’t a problem. It also won’t trigger a warning with --warn-non-bib-fields as it’s being aliased rather than set in the .bib file. The blocks appear in the same order as the corresponding \GlsXtrLoadResources commands. The title for each block is provided in the document using \glsxtrsetgrouptitle.

\documentclass{article}
\usepackage[record,abbreviations,symbols]{glossaries-extra}
\renewcommand{\GlsXtrDefaultResourceOptions}{
  selection={all},src={entries},save-locations={false}}
\GlsXtrLoadResources[type={main},match={entrytype=entry}]
\GlsXtrLoadResources[type={abbreviations},
  match={entrytype=abbreviation}]
\glsxtrsetgrouptitle{pictograph}{Pictographs}
\GlsXtrLoadResources[type={symbols},
  field-aliases={identifier=group},
  match={group=pictograph}]
\glsxtrsetgrouptitle{latin}{Latin Characters}
\GlsXtrLoadResources[type={symbols},
  field-aliases={identifier=group},
1.2 Logical Divisions: type vs group vs parent

\glsxtrsetgrouptitle{greek}{Greek Characters}
\GlsXtrLoadResources[type={symbols},
   field-aliases={identifier=group},
   match={group=greek}]

\begin{document}
\tableofcontents
\printunsrtglossary[type={abbreviations}]
\printunsrtglossary
\printunsrtglossary[type={symbols},style={treegroup}]
\end{document}

In the above example document, the symbols list is divided into three groups, listed in the order: Pictographs, Latin characters and Greek characters. If you want these titles ordered alphabetically then you need a hierarchical structure instead. This can be obtained by aliasing the custom identifier field to parent:

\documentclass{article}
\usepackage[record,stylemods={topic},abbreviations,symbols]{glossaries-extra}
\renewcommand{\GlsXtrDefaultResourceOptions}{
   selection={all},src={entries},save-locations={false}}
\GlsXtrLoadResources[type={main},match={entrytype=entry}]
\GlsXtrLoadResources[type={abbreviations},
   match={entrytype=abbreviation}]
\GlsXtrLoadResources[type={symbols},
   field-aliases={identifier=parent},
   match={entrytype=symbol,entrytype=indexplural}]

\begin{document}
\tableofcontents
\printunsrtglossary[type={abbreviations}]
\printunsrtglossary
\printunsrtglossary[type={symbols},style={topic}]
\end{document}

The style used for the symbols list is now topic rather than treegroup. This results in a slightly different appearance. You can select the most appropriate style according to your needs (see the gallery of predefined styles [15]). The topic ordering is now: Greek characters, Latin characters and Pictographs.
1.3 Defining a New Glossary

Some of the examples in this manual use \texttt{\newglossary*} to define a new glossary type and some use \texttt{\newignoredglossary} or \texttt{\newignoredglossary*}. Why the starred forms and why define an ignored glossary?

The base glossaries package was originally designed to work with \texttt{makeindex}. Support for \texttt{xindy} was later added, but both require three files per glossary type: the transcript file (created by the indexing application), the file written by \texttt{\LaTeX} (and input by the indexing application) and the file input by \texttt{\LaTeX} (and written by the indexing application). So when a new glossary is defined with \texttt{\newglossary}, this not only defines internal control sequences that store the list of entry labels associated with that glossary, the title and the entry format but also has to define internal control sequences that store the three file extensions. The starred form \texttt{\newglossary*} is just a shortcut that forms the extensions from the glossary label. For the purposes of \texttt{bib2gls}, this is simpler than the unstarred version since the extensions are now irrelevant as they are only applicable to \texttt{makeindex} and \texttt{xindy}. (Unless, of course, you are using a hybrid method with \texttt{record=alsoindex}.)

Since some users wanted the ability to define entries that were common enough to not be worth including in any glossary lists, the concept of an ignored glossary was introduced, defined with \texttt{\newignoredglossary}. This only requires an internal control sequence to store the list of entry labels associated with that glossary\footnote{All entries must be assigned to a glossary. If you don’t use the \texttt{type} field the default is used.} and the associated internal command that governs the way that commands like \texttt{\gls} are displayed for that glossary type. Since this type of glossary has no associated files, it can’t be used with \texttt{\printglossary} and therefore isn’t included in the list of glossary labels that’s iterated over by commands like \texttt{\printglossaries}. Since there’s no glossary list (and therefore no targets), \texttt{\newignoredglossary} additionally disables hyperlinks for that glossary type, but it doesn’t disable indexing. The indexing macro is still called, but because there’s no associated file to write to, it has no effect. With \texttt{bib2gls}, the indexing is written to the \texttt{.aux} file and so does have an effect.

Although ignored glossaries can’t be used with \texttt{\printglossary}, they can be used with \texttt{\printunsortedglossary}, which is designed to work without any indexing, but you need to explicitly set the title in the optional argument to override the default. Ignored glossaries still can’t be used in \texttt{\printunsortedglossaries}, since they’re not included in the list that this command iterates over.

So \texttt{\newignoredglossary} (or \texttt{\provideignoredglossary}) is useful with \texttt{bib2gls} if you’re happy to use \texttt{\printunsortedglossary} with the \texttt{type} and \texttt{title} options as it reduces the overall number of internal control sequences. Ignored glossaries are also useful for stand-alone definitions (\texttt{\glsxtrglossentry}) or with \texttt{\printunsortedinnerglossary} as no title is required in those cases (see \texttt{sample-nested.tex} for an example).

Since there is now the possibility of targets (created within \texttt{\printunsortedglossary} or \texttt{\printunsortedinnerglossary} or \texttt{\glsxtrglossentry}), it’s convenient to have an ignored glossary that doesn’t suppress the hyperlinks, which can be obtained with the starred form \texttt{\newignoredglossary*} provided by glossaries-extra (or \texttt{\provideignoredglossary*}).
1.4 Resource Sets

Some resource options, such as master, secondary and trigger-type, need to ensure that a required glossary is defined. In this case, bib2gls uses \provideignoredglossary* in the .glistex file even if --no-provide-glossaries is set. If you haven’t already defined that glossary in the document with \newglossary*, you’ll need to set the title in the optional argument of \printunsrtglossary if you don’t want the default. The glossary won’t be defined on the first run (if the definition is only provided in the .glistex file) but \printunsrtglossary will just give a warning if the type is undefined so it won’t interrupt the document build.

If you want bib2gls to automatically provide unknown glossaries for all entries that have the type field set (unrelated to the master, secondary and trigger-type options) then use the --provide-glossaries switch.

The base glossaries package provides a command that can be used to test the existence of a glossary:

\ifglossaryexists{⟨label⟩}{⟨true⟩}{⟨false⟩}

The unstarred version considers ignored glossaries as non-existent (and so will do ⟨false⟩ for an ignored glossary). As from v4.46, this command now has a starred version \ifglossaryexists* that considers ignored glossaries as existing (and so will do ⟨true⟩ for an ignored glossary). In the event that you have an older version of glossaries, the glossaries-extra package (v1.44+) will provide the starred form if it hasn’t been defined. (In general, it’s best to have up-to-date versions of both glossaries and glossaries-extra.)

1.4 Resource Sets

Each instance of \glsxtrresourcefile or \GlsXtrLoadResources in the document represents a resource set. Each resource set has one or more associated .bib files that provides the data for that set. Command line switches (chapter 3) are applied to all resource sets. Resource options (chapter 5) are only applied to that specific resource set. Each resource set is processed in stages:

**Stage 1 (Initialisation)** Occurs after the .aux file has been read, this stage parses the resource option list and ensures options are valid and don’t cause a conflict. The transcript will show the message

Initialising resource ⟨resource-name⟩

at this point.

**Stage 2 (Parsing)** All the .bib files associated with the resource set are parsed. Entry aliases (identified by entry-type-aliases) are performed. The multi-entry types, such as @bibtexentry and @progenitor, spawn their associated primary entries. Preamble information (provided by @preamble) is saved but is not interpreted at this stage. The transcript will show the message
1.4 Resource Sets

Parsing bib files for resource \textit{(resource-name)}

at this point.

**Stage 3 (Processing Entries)** The transcript will show the message

Processing resource \textit{(resource-name)}

at this point. For each entry that was found in the corresponding set of .bib files:

- Records are transferred to aliases if required (alias-loc).
- Field checks and modifications are performed:
  - field aliases are performed (field-aliases);
  - known fields identified with save-original-id and save-original-entrytype are set (internal fields that don't have a corresponding key for use with \textbackslash newglossaryentry aren't set until the .glistex file is written);
  - ignored fields (identified by ignore-fields) are removed;
  - case-changes (for example, short-case-change) are performed, except for the name field and fields identified with field-case-change;
  - suffixes are appended if required (for example, with short-plural-suffix);
  - field replications are made (replicate-fields), and any of the above case-change or suffixes required on the replicated fields are performed;
  - the group field is assigned if group={\textit{⟨label⟩}} is set;
  - any variables (identified by @string) are expanded (if not already done in any of the previous steps);
  - any fields that have been identified by bibtex-contributor-fields are converted;
  - any fields that have been identified with encapsulate-fields are converted;
  - any fields that have been identified with encapsulate-fields* are converted;
  - any fields that must be converted into a label form (labelify or labelify-list) are processed;
  - any fields identified by dependency-fields are parsed for dependent entries;
  - any fields whose value must be a label are interpreted if interpret-label-fields is set;
  - the parent field is adjusted according to the label prefix settings (label-prefix etc);
1.4 Resource Sets

- \makefirstuc protection is applied according to \_--mfirstuc-protection\_ and \_--mfirstuc-math-protection\_.
- Fields are parsed for commands like \gls or \glshyperlink and also checked for nested links if \_--nested-link-check\_ is set;
- The description field is adjusted according to strip-trailing-nopost;
- End punctuation is checked according to check-end-punctuation;
- Field assignments are made (assign-fields), and any of the above case-change or suffixes required by the destination fields are performed;
- Name adjustment is performed if compound-adjust-name is set (and the criteria is met);
- Name case-change is performed if name-case-change is set;
- If copy-alias-to-see={true} the alias is copied to the see field;
- General field case changes identified by field-case-change are performed;
- Any fields that have been identified with interpret-fields are replaced with their interpreted values;
- Any fields that have been identified with hex-unicode-fields will have Unicode characters replaced;
- Check for nonumberlist.

- The dual version (if appropriate) is created.
- Records are added to the entry’s location list (or transferred to the dual/primary according to combine-dual-locations).
- The type, category and counter fields are set according to type, dual-type, category, dual-category, counter and dual-counter.
- Filtering is applied (according to options like match but not selection or limit).
- Required fields are checked for existence.
- Dependencies are registered (if selection={recorded and deps} or selection ={recorded and deps and see}).
- Any fields that have been identified by date-time-fields, date-fields or time-fields are converted.

If selection={recorded and deps and see} then any recorded entries that have been cross-referenced by an unrecorded entry, will register a dependency with the unrecorded entry.

The compound entry options compound-dependent and compound-add-hierarchy are implemented, if enabled.

Finally, supplemental records are added to entries.
1.4 Resource Sets

**Stage 4 (Selection, Sorting, Writing)** Entries are selected from the list according to the `selection` setting, sorting is performed (if required), truncation is applied (if `limit` is set) and the `.glistex` file is written. The transcript will show the message

```
Selecting entries for resource ⟨resource-name⟩
```

or (if `master`)

```
Processing master ⟨resource-name⟩
```

at this point.

Parent entries must always be in the same resource set as their child entries. (They may be defined in different `.bib` files as long as all those `.bib` files are listed in the same `src`.) Other forms of dependencies may be in a different resource set under certain circumstances. These types of dependencies are instances of commands such as \gls being found (for example, in the `description` field), or the cross-reference fields (`see`, `seealso` or `alias` or fields identified with `dependency-fields`) in recorded entries that reference unrecorded entries.

The “cross-referenced by” dependencies enabled with `selection={recorded and deps and see}` (where an unrecorded entry references a recorded entry through the cross-reference fields) aren’t supported across resource sets (even with `--force-cross-resource-refs`).

A cross-resource reference is a reference from a recorded entry provided in one resource set to an unrecorded entry in another resource set. Since the contents of each resource set’s preamble must be processed before fields can be interpreted and one resource set’s preamble may contain definitions that override another, cross-resource references can’t be supported if fields containing cross-referencing information need to be interpreted.

The cross-resource reference mode determines whether or not `bib2gls` can support cross-resource references. If enabled, the message

```
Cross-resource references allowed.
```

will be written to the transcript otherwise the message is

```
Cross-resource references disabled.
```

The mode can only be enabled if the following condition is satisfied:

- the interpreter is off (`--no-interpret`), or
- every resource set either doesn’t have a preamble (`@preamble`) or has interpret `-preamble={false}` set.

If you know the preamble contents won’t cause a problem, you can force the cross-resource references mode on with `--force-cross-resource-refs`.

If you don’t use either `selection={recorded and deps}` or `selection={recorded and deps and see}` then the dependencies aren’t picked up for that resource set (and so can’t be cross-referenced from another resource set).
1.5 bib2gls Quarks

Trails don’t work with cross-resource references. For example, if entry $A$ has been recorded and depends on entry $B$ that hasn’t been recorded, then $B$ can be picked up from a different resource set, but if $A$ and $B$ are in the same resource set and $B$ is dependent on $C$ which is in a different resource set then $C$ won’t be picked up if it hasn’t been recorded because $B$ hasn’t been recorded and is in a different resource set.

If the cross-resource reference mode is enabled then stage 3 and stage 4 are processed in separate loops, otherwise they are processed in the same loop.

1.5 bib2gls Quarks

A bib2gls quark is similar in principle to a \TeX\ quark, in that it is a token that looks like a control sequence but isn’t intended to be interpreted as a \TeX command. Unlike \TeX quarks, their name isn’t prefixed with \texttt\verb|\_| and can coincidentally look the same as a \TeX command. This is particularly the case with regular expressions that have escaped characters to indicate a literal character. For example, in a regular expression a pipe or vertical bar character $|$ indicates “or”. If you want to match a literal pipe, you need to identify this with $\backslash$. This is distinct from, but visually identical to, the \TeX command used to create a double vertical bar in maths mode.

The resource options provided in \texttt\verb|\GlsXtrLoadResources| expand as they are written to the \texttt{.aux} file. This allows commands to be used within the resource options that expand to a complex option that may be required multiple times. For example, \texttt\verb|\GlsXtrBibTeX-EntryAliases| or \texttt\verb|\glsxtrhyphenrules|. Unfortunately, this means that quarks must be prevented from expansion as they form part of the option syntax and are not intended for use in the document.

This means that, unless they happen to coincidentally be robust commands, they must be preceded by either \texttt\verb|\protect| or \texttt\verb|\string|. Since \texttt\verb|\protect| adds a space afterwards, \texttt\verb|\string| is usually better if the syntax requires that spaces after quarks are significant.

This can lead to cumbersome expressions, but it’s possible to redefine \texttt\verb|\glsxtrresourceinit| to locally redefine these quarks to expand to detokenized forms of themselves. For example:

\begin{verbatim}
\renewcommand*{\texttt\verb|\glsxtrresourceinit|}{\let\u\glshex}
\end{verbatim}

Since there are a number of these quarks, as from v1.51, glossaries-extra-bib2gls (which is automatically loaded with \texttt\verb|record|) provides \texttt\verb|\GlsXtrResourceInitEscSequences|, so you can change the above to the following:

\begin{verbatim}
\renewcommand*{\texttt\verb|\glsxtrresourceinit|}{% 
 \GlsXtrResourceInitEscSequences
}
\end{verbatim}

This will locally define the quarks listed below. Since \texttt\verb|\glsxtrresourceinit| is scoped so that it only has an effect within the protected write, this shouldn’t interfere with the corresponding commands that are required in the document. Note that these quarks should only be used in their designated contexts.
1.6 Indexing

**General** \u(\text{hex}) is recognised in certain resource options (such as \texttt{field-concat-sep}) as indicating the Unicode character with the given hexadecimal code.

**Regular expressions** The following indicate a literal character: \. \| \^ \& \+ \< \> \* \$ \~ \- \( \) \[ \] " \- \? \: \#. Note that regular expressions in resource options are typically anchored, so there shouldn’t be any need to use \^ or $ to denote the start and end.

**Field assignments** The following commands may be used in the \langle\texttt{element-list}\rangle syntax of \texttt{assign-fields}: \texttt{\CS}, \texttt{\MGP}, \texttt{\LEN}, \texttt{\TRIM}, \texttt{\INTERPRET}, \texttt{\LC}, \texttt{\UC}, \texttt{\FIRSTLC}, \texttt{\FIRSTUC}, and \texttt{\TITLE}.

**Conditionals** The \langle\texttt{condition}\rangle part of the \texttt{assign-fields} syntax recognises \texttt{\LEN}, \texttt{\CAT}, \texttt{\IN}, \texttt{\NIN}, \texttt{\PREFIXOF}, \texttt{\NOTPREFIXOF}, \texttt{\SUFFIXOF}, \texttt{\NOTSUFFIXOF} and \texttt{\NULL}.

Finally, this isn’t actually a quark, but \texttt{\cs{\langle\texttt{csname}\rangle}} is defined to expand to the literal string \texttt{\csname} so you can use it for any other escape sequences that aren’t covered above. For example, \texttt{\cs{n}} for a newline \texttt{\n}.

### 1.6 Indexing

The dual index entries such as @\texttt{dualindexentry} (described in section 4.7) are designed to provide a way of including an entry in a glossary (with a description) and also include the term (without the description) in an index. Additional terms that should only appear in the index can be defined with \texttt{@index}. (See, for example, the \texttt{sample-multi1.tex} and \texttt{sample-multi2.tex} sample files.)

Although \texttt{bib2gls} is designed to create indexes as well as glossary lists using the same interface (\texttt{\gls} etc), it is possible to have a mixture of \texttt{bib2gls} and \texttt{\index}. For example:

\begin{verbatim}
\documentclass{report}
\usepackage{makeidx}
\usepackage[record]{glossaries-extra}
\makeindex
\GlsXtrLoadResources[src={entries}]
\glssetcategoryattribute{general}{dualindex}{true}
\glssetcategoryattribute{symbol}{dualindex}{true}
\glssetcategoryattribute{abbreviation}{dualindex}{true}
\glssetcategoryattribute{general}{indexname}{hyperbf}
\glssetcategoryattribute{symbol}{indexname}{hyperbf}
\glssetcategoryattribute{abbreviation}{indexname}{hyperbf}
\end{verbatim}
1.7 Security

The document is called myDoc.tex then the document build is:

```latex
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
makeindex myDoc.idx
pdflatex myDoc
```

This requires an additional \TeX{} call between `bib2gls` and `makeindex` since the entries must be defined before they can be indexed (and they can’t be defined until `bib2gls` creates the associated `.glstex` files).

Note that this method will use the `sort` value obtained by `bib2gls` as the `<sort>` part within `\index{<sort>@<actual>}`. Be careful if you use `makeindex` as this can result in Unicode characters appearing in the sort value, which `makeindex` doesn’t support. The `<actual>` part is given by `\glsentryname{<label>}`. (You can change the `<sort>` and `<actual>` parts by redefining `\glsxtraautoindexassignsor` and `\glsxtraautoindexentry`. See the glossaries-extra manual for further details.)

1.7 Security

\TeX{} Live come with security settings `openin_any` and `openout_any` that, respectively, govern read and write file access (in addition to the operating system’s file permissions). `bib2gls` uses `kpsewhich` to determine these values and honours them. MiKTeX doesn’t use these settings, so if these values are unset, `bib2gls` will default to `a` (any) for `openin_any` and `p` (paranoid) for `openout_any`.

The only external processes that are run by `bib2gls` are calls to `kpsewhich` to check the security settings and locate files on \TeX{}’s path. These are started with Java’s `ProcessBuilder` class so there should be no issues with spaces or shell special characters in the argument. The `--debug` switch will write the process call in the transcript file and will delimit the argument in the log with single quote characters for convenience, but the process isn’t actually called in that way.

`bib2gls` creates files with the extension `.glstex`, which are input by `\glsxtrresourcefile` (and therefore by the shortcut `\GlsXtrLoadResources`). This extension is fixed and is imposed by both `bib2gls` and `\glsxtrresourcefile`. `bib2gls` also creates a transcript file with the default extension `.glg`. This may be overridden by the `--log-file` switch, but `bib2gls` always forbids write access to any file with the following extensions: `.tex`, `.ltx`, `.sty`, `.cls`, `.bib`, `.dtx`, `.ins`, `.def` and `.ldf`.
1.8 Localisation

The messages produced by \bib2gls are fetched from a resource file called \bib2gls-\langle lang\rangle.xml, where \langle lang\rangle is a valid Internet Engineering Task Force (IETF) language tag.

The appropriate file is searched for in the following order, where \langle locale\rangle is the Java locale or the value supplied by the --locale switch:

1. \langle lang\rangle exactly matches \langle locale\rangle. For example, my locale is en-GB, so \bib2gls will first search for \bib2gls-en-GB.xml. This file doesn’t exist, so it will try again.

2. If \langle locale\rangle has an associated script, the next try is with \langle lang\rangle set to \langle lang code\rangle-\langle script\rangle where \langle lang code\rangle is the two letter ISO language code and \langle script\rangle is the script code. For example, if \langle locale\rangle is sr-RS-Latn then \bib2gls will search for \bib2gls-sr-sr-Latn.xml if \bib2gls-sr-RS-Latn.xml doesn’t exist.

3. The final attempt is with \langle lang\rangle set to just the two letter ISO language code. For example, \bib2gls-sr.xml.

If there is no match, \bib2gls will fallback on the English resource file \bib2gls-en.xml. (Currently only \bib2gls-en.xml exists as my language skills aren’t up to translating it. Any volunteers who want to provide other language resource files would be much appreciated.)

In addition to the main language file, it’s possible to have supplementary files that provide text that matches the resource locale. These are in files called \bib2gls-extra-\langle lang\rangle.xml, which has the same format as \bib2gls-\langle lang\rangle.xml. These supplementary files will be loaded automatically if they exist and if you have glossaries-extra v1.5i+ (which will save a list of all tracked languages for the document).

Note that if you use the loc-prefix={true} option, the textual labels (“Page” and “Pages” in English) will be first be attempted from the supplementary file with the tags tag.\langle lang\rangle.page and tag.\langle lang\rangle.pages (where \langle lang\rangle is the language code) and then, if not found, from the main resource file using the tags tag.page and tag.pages. In the event that the loaded resource file doesn’t match the document language and there’s no supplementary file, you will have to manually set the correct translation (in English, this would be loc-prefix={Page, Pages}). The default definition of \bibglspassim is also obtained from the resource file in a similar manner.

There are also keys in the resource file to assist case-conversion. Currently, there’s only support for the Dutch “IJ” case.

1.9 Conditional Document Build

If you are using a document build method that tries to determine whether or not \bib2gls should be run, you can find the information by searching the .aux file for instances of \glsxtr@resource{\langle options\rangle}{\langle filename\rangle}
1.10 Manual Installation

Each instance corresponds to an instance of \glsxtrresourcefile where \textit{filename} is the base name of the .glstex file that \texttt{bib2gls} needs to create for this resource set. If the \textit{options} part is missing the \texttt{src} option, then \textit{filename} also indicates the base name for the .bib file.

So the simplest check to determine if \texttt{bib2gls} needs to be run is to test if the .aux file contains \texttt{\glsxtr@resource}. For example, with \texttt{arara} version 4.0:

\%
\texttt{arara: bib2gls if found("aux", "\glsxtr@resource")}

A sophisticated method could check if \texttt{\textit{filename}.glstex} is missing or is older than the document .tex file for each instance of \texttt{\glsxtr@resource} found in the .aux file.

It might also be possible, although far more complex, to parse the \textit{options} part in each instance of \texttt{\glsxtr@resource} for \texttt{src} and determine if the corresponding .bib file or files are newer than the .tex file.

It’s not possible to determine if the location lists require updating, just as it’s not possible to do this for the table of contents (toc), list of figures, list of tables etc. (Or, if it could be implemented, the required code would make the document build far more complicated.)

In general, the basic algorithm is:

1. Run \LaTeX{} (or PDF\LaTeX{} etc).

2. If \texttt{\glsxtr@resource} is found in the .aux file then:
   a) run \texttt{bib2gls};
   b) run \LaTeX{} (or PDF\LaTeX{} etc).

3. If \texttt{@istfilename} is found in the .aux file then:
   a) run \texttt{makeglossaries} (or \texttt{makeglossaries-lite});
   b) run \LaTeX{} (or PDF\LaTeX{} etc).

This allows for the \texttt{record=\{alsoindex\}} package option. See also “Incorporating \texttt{makeglossaries} or \texttt{makeglossaries-lite} or \texttt{bib2gls} into the document build” [12].

1.10 Manual Installation

If you are unable to install \texttt{bib2gls} through your \TeX{} package manager, you can install manually using the instructions below. Replace \texttt{\TEXMF{}} with the path to your local or home TEXMF tree (for example, \texttt{/\textasciitilde texmf}).

Copy the files provided to the following locations:

- \texttt{\TEXMF{}/scripts/bib2gls/bib2gls.jar} (Java application.)
- \texttt{\TEXMF{}/scripts/bib2gls/convertgls2bib.jar} (Java application.)
- \texttt{\TEXMF{}/scripts/bib2gls/texparserlib.jar} (Java library.)
1.10 Manual Installation

- ⟨TEXMF⟩/scripts/bib2gls/resources/bib2gls-en.xml (English resource file.)
- ⟨TEXMF⟩/doc/support/bib2gls/bib2gls.pdf (This document.)
- ⟨TEXMF⟩/doc/support/bib2gls/bib2gls-begin.pdf (Introductory guide.)

If you use the Unix `man` command, copy the `bib2gls.1` and `convertgls2bib.1` files to the appropriate location.

If you are using a Unix-like system, there are also bash scripts provided called `bib2gls.sh` and `convertgls2bib.sh`. Either copy then directly to somewhere on your path without the `.sh` extension, for example:

```bash
cp bib2gls.sh ~/bin/bib2gls
cp convertgls2bib.sh ~/bin/convertgls2bib
```

or copy the files to ⟨TEXMF⟩/scripts/bib2gls/ and create a symbolic link to them called just `bib2gls` and `convertgls2bib` from somewhere on your path, for example:

```bash
cp bib2gls.sh ~/texmf/scripts/bib2gls/
cp convertgls2bib.sh ~/texmf/scripts/bib2gls/
cd ~/bin
ln -s ~/texmf/scripts/bib2gls/bib2gls.sh bib2gls
ln -s ~/texmf/scripts/bib2gls/convertgls2bib.sh convertgls2bib
```

The `texparserlib.jar` file isn’t an application but is a library used by both `bib2gls.jar` and `convertgls2bib.jar`, and so needs to be in the same class path. (The library is in a separate GitHub repository [10] as it’s also used by some of my other applications.)

Windows users can create a `.bat` file that works in a similar way to the bash scripts. To do this, create a file called `bib2gls.bat` that contains the following:

```bash
@ECHO OFF
FOR /F "tokens=" %%I IN ('kpsewhich --progname=bib2gls --format=texmfscripts bib2gls.jar') DO SET JARPATH=%%I
java -Djava.locale.providers=CLDR,JRE -jar "%JARPATH%" %*
```

Save this file to somewhere on your system’s path. (Similarly for `convertgls2bib`.) Note that TeX distributions for Windows usually convert .jar files to executables.

You may need to refresh TeX’s database to ensure that `kpsewhich` can find the .jar files. To test that the application has been successfully installed, open a command prompt or terminal and run the following command:

```bash
bib2gls --version
convertgls2bib --version
```

This should display the version information for both applications.
2 TeX Parser Library

The bib2gls application requires the TeX Parser Library texparserlib.jar\(^1\) which is used to parse the .aux and .bib files.

With the --interpret switch on (default), this library is also used to interpret the sort value when it contains a backslash \ or a tilde ~ or a dollar symbol $ or braces \{ \} (and when the sort option is not unsrt or none or use).\(^2\)

The other cases that the interpreter is used for are:

- when set-widest is used to determine the width of the name field;
- if labelify or labelify-list are set the identified field values are first interpreted (if they contain \ { } - or $) before being converted to labels;
- if interpret-label-fields={true} is set and the parent, category, type, group, seealso or alias fields contain \ or \{ or \} the interpreter is used since these fields must be just a label (other special characters aren’t checked as they won’t expand to characters allowed in a label).

Information in the .aux file is parsed for specific commands but the arguments of those commands are not interpreted so, for example, UTF-8 characters that occur in any resource options will need to be detokenized when using inputenc to prevent expansion when they are written to the .aux file. (In some options, such as sort-rule, you can use \glshex\{hex\} syntax to specify a UTF-8 character.) Note that newer \LaTeX kernels have better support for UTF-8 and this issue is less likely to occur.

The --no-interpret switch will turn off the interpreter, but the library will still be used to parse the .aux and .bib files. Note that the see field doesn’t use the interpreter with interpret-label-fields={true} as it may legitimately contain \LaTeX code in the optional tag part (such as \seealsoname or \alsoname).

The parser has a different concept of expansion to \TeX and will expand some things that aren’t expanded by \LaTeX (such as \MakeUppercase and \char) and won’t expand other commands that would be expanded by \LaTeX (such as commands defined in terms of complicated internals).

If you get a StackOverflowError while a field is being interpreted (with a long stack trace that contains repeated file names and line numbers) then it’s likely you have an infinite loop. For example, this can be triggered if a field contains \foo that has been defined as:

\(^1\)https://github.com/nlct/texparser

\(^2\)The other special characters are omitted from the check: the comment symbol % is best avoided in field values, the subscript and superscript characters _ and ^ should either be encapsulated by $ or by \ensuremath, which will be picked up by the check for $ or \\, and the other special characters would indicate something too complex for the interpreter to handle.
This will obviously also cause an error in the \LaTeX{} document as well (unless the document has a different definition that doesn't have this unbounded recursion).

The `texparserlib.jar` library is not a \LaTeX{} engine and there are plenty of situations where it doesn't work. In particular, in this case it's being used in a fragmented context without knowing most of the packages used by the document or any custom commands or environments provided within the document.

`bib2gls` can detect from the log file a small number of packages that the parser recognizes. Note that in some cases there's only very limited support. For example, `siunitx`'s `\si` command is recognized but other commands from that package aren't. See `--list-known-packages` (page 36) for further details.

Since the parser doesn't have a full set of commands available within the \LaTeX{} document, when it encounters `\renewcommand` it won't check if the command is undefined. If the command isn't defined, it will simply behave like `\newcommand`. Whereas with `\providecommand` the parser will only define the command if it's unrecognized.

The interpreter has its own internal implementation of the glossary-related commands listed in table 2.1. These may be overridden by custom packages provided with the `--custom-packages` switch. Note that commands that reference an entry, such as `\glsentryname`, aren't guaranteed to work across resource sets and will only be able to look up field values that are known to `bib2gls`. (For example, the `name` field for abbreviations is typically set by `bib2gls`.)

If a command isn't recognized, you can provide it in the \texttt{@preamble} and use `\char` to map a symbol to the most appropriate Unicode character. For example, suppose your document loads a package that provides symbols for use on maps, such as `Harbour`, `Battlefield` and `Stadium`, then you can provide versions of these commands just for `bib2gls`'s use:

\begin{verbatim}
@preamble{"\providecommand{\Harbour}{\char"2693}
\providecommand{\Battlefield}{\char"2694}
\providecommand{\Stadium}{\char"26BD}"}
\end{verbatim}

Since these use `\providecommand`, they won’t overwrite the document’s version (provided these commands have been defined before `\GlsXtrLoadResources`). Alternatively, you can instruct `bib2gls` to not write the \texttt{@preamble} contents to the resource file using `write-preamble={false}`. Now you can either sort these symbols by their Unicode values (\texttt{sort={letter-case}}) or provide a custom rule that recognizes these Unicode characters (for example, \texttt{sort={custom},sort-rule={\glshex2694 < \glshex2693 < \glshex26BD}}).

\LaTeX{} syntax can be quite complicated and, in some cases, far too complicated for simple regular expressions. The \LaTeX{} parser library performs better than a simple pattern match, and that's the purpose of `texparserlib.jar` and why it's used by `bib2glsl` (and by `convert-gls2bib`). When the `--debug` mode is on, any warnings or errors triggered by the interpreter will be written to the transcript prefixed with `texparserlib:` (the results of the conversions will be included in the transcript as informational messages prefixed with `texparserlib:` even with `--no-debug`).

\footnote{These commands won’t work with PDF\LaTeX{}, as the `\char` values are too large, but they’re fine for `bib2glsl`.}
Table 2.1: Glossary-Related Commands Implemented by the bib2gls Interpreter

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>\bibglsampersandchar</td>
<td>\bibglscircumchar</td>
<td>\bibglscontributor</td>
</tr>
<tr>
<td>\bibglscontributorlist</td>
<td>\bibglsdate</td>
<td>\bibglsdatetime</td>
</tr>
<tr>
<td>\bibglsdollarchar</td>
<td>\bibglsfirstuc</td>
<td>\bibglshashchar</td>
</tr>
<tr>
<td>\bibglshyperlink</td>
<td>\bibglslowercase</td>
<td>\bibglstime</td>
</tr>
<tr>
<td>\bibglstitlecase</td>
<td>\bibglsunderscorechar</td>
<td>\bibglsuppercase</td>
</tr>
<tr>
<td>\glsbackslash</td>
<td>\glsclosebrace</td>
<td>\glsentryfirst</td>
</tr>
<tr>
<td>\Glsentryfirst</td>
<td>\glsentryfirstplural</td>
<td>\Glsentryfirstplural</td>
</tr>
<tr>
<td>\glsentrylong</td>
<td>\Glsentrylong</td>
<td>\glsentrylongpl</td>
</tr>
<tr>
<td>\Glsentrylongpl</td>
<td>\glsentryname</td>
<td>\Glsentryname</td>
</tr>
<tr>
<td>\glsentryplural</td>
<td>\Glsentryplural</td>
<td>\glsentryshort</td>
</tr>
<tr>
<td>\Glsentryshort</td>
<td>\glsentryshortpl</td>
<td>\Glsentryshortpl</td>
</tr>
<tr>
<td>\glsentriesymbol</td>
<td>\Glsentriesymbol</td>
<td>\glsentriesymbolplural</td>
</tr>
<tr>
<td>\Glsentriesymbolplural</td>
<td>\glsentrytext</td>
<td>\Glsentrytext</td>
</tr>
<tr>
<td>\glsentrytitlecase</td>
<td>\glsentryuservi</td>
<td>\Glsentryuservi</td>
</tr>
<tr>
<td>\glsentryuserii</td>
<td>\glsentryuserii</td>
<td>\glsentryuserii</td>
</tr>
<tr>
<td>\Glsentryuseri</td>
<td>\glsentryuseriv</td>
<td>\Glsentryuseriv</td>
</tr>
<tr>
<td>\glsentryuserv</td>
<td>\glsentryuserv</td>
<td>\glsentryuserv</td>
</tr>
<tr>
<td>\Glsentryuservi</td>
<td>\glshyperlink</td>
<td>\glsopenbrace</td>
</tr>
<tr>
<td>\glspercentchar</td>
<td>\glstildechar</td>
<td>\GlsXtrEnableInitialTagging</td>
</tr>
<tr>
<td>\glsxthiername</td>
<td>\Glsxthiername</td>
<td>\Glsxthiername</td>
</tr>
<tr>
<td>\GLSXthiernamesep</td>
<td>\glsxthiernamesep</td>
<td></td>
</tr>
<tr>
<td>\glsxtrprovidecommand</td>
<td>\glsxtrustorousfield</td>
<td>\Glsxtrustorousfield</td>
</tr>
<tr>
<td>\GLSXtrustorousfield</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For example, suppose the \texttt{.bib} file includes:

\begin{verbatim}
@preamble{
  "\providecommand{\mtx}[1]{\boldsymbol{#1}}
  \providecommand{\set}[1]{\mathcal{#1}}
  \providecommand{\card}[1]{|\set{#1}|}
  \providecommand{\imaginary}{i}"
}

@entry{M, 
  name={\{}\mtx{M}\}, 
  text={\mtx{M}},
  description={a matrix}
}

@entry{v, 
  name={\{}\vec{v}\},
  text={\vec{v}},
  description={a vector}
}

@entry{S, 
  name={\{}\set{S}\},
  text={\set{S}},
  description={a set}
}

@entry{card, 
  name={\{}\card{S}\},
  text={\card{S}},
  description={the cardinality of the set \set{S}}
}

@entry{i, 
  name={\{}\imaginary\},
  text={\imaginary},
  description={square root of minus one (\$\sqrt{-1}$)}
}
\end{verbatim}

(The empty group at the start of the \texttt{name} fields protects against the possibility that the gloss-name category attribute might be set to \texttt{firstuc}, which automatically converts the first letter of the name to upper case when displaying the glossary. See also \texttt{--mfirstuc-protection} and \texttt{--mfirstuc-math-protection}.)

None of these entries have a \texttt{sort} field so the \texttt{name} is used (see section 5.8). If the entry type had been \texttt{@symbol} instead, the fallback would be the entry’s label. This means that with
@symbol instead of @entry, and the default sort-field={sort}, and with sort={letter-case}, these entries will be defined in the order: M, S, card, i, v (since this is the case-sensitive letter order of the labels) whereas with sort-field={letter-nocase}, the order will be: card, i, M, S, v (since this is the case-insensitive letter order of the labels).

However, with @entry, the fallback field will be taken from the name which in the above example contains \TeX\ code, so \texttt{bib2gls} will use \texttt{texparserlib.jar} to interpret this code. The library has several different ways of writing the processed code. For simplicity, \texttt{bib2gls} uses the library’s HTML output and then strips the HTML markup and trims any leading or trailing spaces. The library method that writes non-ASCII characters using “&x⟨hex⟩;” markup is overridden by \texttt{bib2gls} to just write the actual Unicode character, which means that the letter-based sorting options will sort according to the integer value ⟨hex⟩ rather than the string “&x⟨hex⟩;”.

The interpreter is first passed the code provided with @preamble:

\begin{verbatim}
\providecommand{\mtx}{[1]{\boldsymbol{#1}}} \\
\providecommand{\set}{[1]{\mathcal{#1}}} \\
\providecommand{\card}{[1]{|\set{#1}|}} \\
\providecommand{\imaginary}{i}
\end{verbatim}

(Unless \texttt{interpret-preamble=false}). This means that the provided commands are now recognised by the interpreter when it has to parse the fields later.

In the case of the M entry in the example above, the code that’s passed to the interpreter is:

\begin{verbatim}
{}\mtx{M}$
\end{verbatim}

The transcript (.g1g) file will show the results of the conversion:

\begin{verbatim}
texparselib: {}\mtx{M}$ -&gt; M
\end{verbatim}

So the sort value for this entry is set to “M”. The font change (caused by math-mode and \boldsymbol) has been ignored. The sort value therefore consists of a single Unicode character 0x4D (Latin upper case letter “M”, decimal value 77).

For the v entry, the code is:

\begin{verbatim}
{}\vec{v}$
\end{verbatim}

The transcript shows:

\begin{verbatim}
texparselib: {}\vec{v}$ -&gt; \vec{v}
\end{verbatim}

So the sort value for this entry is set to “\vec{v}”, which consists of two Unicode characters 0x76 (Latin lower case letter “v”, decimal value 118) and 0x20D7 (combining right arrow above, decimal value 8407).

For the set entry, the code is:

\begin{verbatim}
{}\set{S}$
\end{verbatim}
The transcript shows:

texparserlib: {}$\set{S}$ $\rightarrow$ S

So the sort value for this entry is set to “S” (again ignoring the font change). This consists of a single Unicode character 0x53 (Latin upper case letter “S”, decimal value 83).

For the card entry, the code is:

{}$\card{S}$

The transcript shows:

texparserlib: {}$\card{S}$ $\rightarrow$ |S|

So the sort value for this entry is set to “|S|” (the | characters from the definition of \card provided in @preamble have been included, but the font change has been discarded). In this case the sort value consists of three Unicode characters 0x7C (vertical line, decimal value 124), 0x53 (Latin upper case letter “S”, decimal value 83) and 0x7C again. If interpret-preamble={false} had been used, \card wouldn’t be recognised and would be discarded leaving just “S” as the sort value.

(Note that if \vert is used instead of | then it would be converted into the mathematical operator 0x2223 and result in a different order.)

For the i entry, the code is:

{}$\imaginary$

The transcript shows:

texparserlib: {}$\imaginary$ $\rightarrow$ i

So the sort value for this entry is set to “i”. If interpret-preamble={false} had been used, \imaginary wouldn’t be recognised and would be discarded, leaving an empty sort value.

This means that in the case of the default sort-field={sort} with sort={letter-case}, these entries will be defined in the order: M (M), S (S), i (i), \v (\v) and card (|S|). In this case, the entries have been sorted according to the character codes. If you run bib2gls with --verbose the decimal character codes will be included in the transcript. For this example:

i $\rightarrow$ ‘i’ [105]
card $\rightarrow$ ‘|S|’ [124 83 124]
M $\rightarrow$ ‘M’ [77]
S $\rightarrow$ ‘S’ [83]
v $\rightarrow$ ‘\v’ [118 8407]

The --group option (in addition to --verbose) will place the letter group in parentheses before the character code list.
i -> 'i' (i) [105]
card -> '|S|' [124 83 124]
M -> 'M' (M) [77]
S -> 'S' (S) [83]
v -> 'v' (v) [118 8407]

(Note that the card entry doesn’t have a letter group since the vertical bar character isn’t considered a letter.)

If sort={letter-nocase} is used instead then, after conversion by the interpreter, the sort values will all be changed to lower case. The order is now: i (i), M (M), S (S), v (v) and card (|S|). The transcript (with --verbose) now shows

i -> 'i' [105]
card -> '|s|' [124 115 124]
M -> 'm' [109]
S -> 's' [115]
v -> 'v' [118 8407]

With --group (in addition to --verbose) the letter groups are again included:

i -> 'i' (I) [105]
card -> '|s|' [124 115 124]
M -> 'm' (M) [109]
S -> 's' (S) [115]
v -> 'v' (V) [118 8407]

Note that the letter groups are upper case not lower case. Again the card entry doesn’t have an associated letter group.

If a locale-based sort is used, the ordering will follow the locale’s alphabet rules. For example, with sort={en} (English, no region or variant), the order becomes: card (|S|), i (i), M (M), S (S) and v (v). The transcript (with --verbose) shows the collation keys instead:

i -> 'i' [0 92 0 0 0 0]
card -> '|s|' [0 66 0 102 0 66 0 0 0 0]
M -> 'M' [0 96 0 0 0 0]
S -> 'S' [0 102 0 0 0 0]
v -> 'v' [0 105 0 0 0 0]

Again the addition of the --group switch will show the letter groups.4

Suppose I add a new symbol to my .bib file:

@symbol{angstrom,
      name={\AA},
      description={\AA ngstr"om}}

4For more information on collation keys see the CollationKey class in Java’s API [2].
and I also use this entry in the document. Then with `sort={en}`, the order is: card (`|S|`), angstrom (`Å`), i (`i`), M (`M`), S (`S`), and v (`v`). The `--group` switch shows that the angstrom entry (`Å`) has been placed in the “Å” letter group.

However, if I change the locale to `sort={sv}`, the angstrom entry is moved to the end of the list and the `--group` switch shows that it’s been placed in the “Å” letter group.

If you are using Java 8, you can set the `java.locale.providers` property [8] to use the Unicode Common Locale Data Repository (CLDR) locale provider, which has more extensive support for locales than the native JRE. For example:

```java
java.locale.providers=CLDR,JRE
```

This should be enabled by default for Java 9. The property can either be set in a script that runs bib2gls, for example,

```java
java -Djava.locale.providers=CLDR,JRE,SPI -jar "$jarpath" "$@"
```

(where `$jarpath` is the path to the `bib2gls.jar` file and "$@" is the argument list) or you can set the property as the default for all Java applications by adding the definition to the `JAVA_TOOL_OPTIONS` environment variable [9]. For example, in a bash shell:

```bash
export JAVA_TOOL_OPTIONS='-Djava.locale.providers=CLDR,JRE,SPI'
```

or in Windows:

```bash
set JAVA_TOOL_OPTIONS=-Djava.locale.providers=CLDR,JRE,SPI
```

---

5 A better method is to use `siunitx` instead.
3 Command Line Options

The syntax of bib2gls is:

```
bib2gls [(options)] (filename)
```

where `(filename)` is the name of the `.aux` file. (The extension may be omitted.) Only one `(filename)` is permitted. Available options are listed below.

### 3.1 Common Options

**--help (or --h)**
Display the help message and quit.

**--version (or --v)**
Display the version information and quit. As from v2.5, this now includes the version number of the `texparserlib.jar` library.

**--debug [⟨n⟩]**
Switch on debugging mode. If ⟨n⟩ is present, it must be a non-negative integer indicating the debugging level. If omitted 1 is assumed. This option also switches on the verbose mode. A value of 0 is equivalent to **--no-debug**.

**--no-debug (or --nodebug)**
Switches off the debugging mode.

**--verbose**
Switches on the verbose mode. This writes extra information to the terminal and transcript file.

**--no-verbose (or --noverbose)**
Switches off the verbose mode. This is the default behaviour. Some messages are written to the terminal. To completely suppress all messages (except errors), switch on the silent mode. For additional information messages, switch on the verbose mode.
--quiet (or -q)
Suppresses all messages except for errors that would normally be written to the terminal. Warnings and informational messages are written to the transcript file, which can be inspected afterwards.

--silent
Synonym of --quiet.

--locale ⟨lang⟩ (or -l ⟨lang⟩)
Specify the preferred language resource file, where ⟨lang⟩ is a valid IETF language tag. This option requires an appropriate bib2gls-⟨lang⟩.xml resource file otherwise bib2gls will fallback on English. This also sets the default document locale when the doc keyword (in options such as sort={doc}) is used and the document doesn’t have any language support. Note that the locale keyword (in options such as sort={locale}) uses the Java locale and is not governed by this switch.

If a document doesn’t have any locale support or has support for more than one language then it’s best to explicitly set the required locale in the appropriate resource set using the locale resource option, to specify the default resource locale, or set the locale for individual options, such as sort.

--group (or -g)
The glossaries-extra record package option automatically creates a new internal field called group. If the --group switch is used with the default group={auto} option then, when sorting, bib2gls will try to determine the group for each entry and assign it to the group field. (Some sort options ignore this setting.) This value will be picked up by \print-unrstglossary if group headings are required (for example with the indexgroup style) or if group separators are required (for example, the index style with the default nogroups=skip ={false}). If you don’t require grouping within the glossary, there’s no need to use this switch. Note that this switch doesn’t automatically select an appropriate glossary style.

If you want sub-groups, you will need to use the group-level resource option and ensure you have glossaries-extra v1.49+. Small groups can be merged with the merge-small-groups resource option.
3.1 Common Options

The `group` field should typically not be set in the `.bib` file and will trigger a warning if found. The explicit use of the `group` key will override `bib2gls`'s normal group formation behaviour, which can cause unexpected results. The custom use of the `group` field requires some care. As a general rule, if you find yourself wanting to use the `group` field in the `.bib` file, then the chances are that what you actually have is a hierarchical glossary (list of topics) and what you really need is the `parent` field. Compare the example files `sample-textsymbols.tex` and `sample-textsymbols2.tex`. See also section 1.2.

There are eight types of groups:

- **letter group** The first non-ignored character of the sort value is alphabetic. This type of group occurs when using the alphabetic sort methods listed in table 5.2 or with the letter sort methods listed in table 5.3 or with the letter-number sort methods listed in table 5.4. The group label is obtained from `\bibglslettergroup`.

- **non-letter group (or symbol group)** The first non-ignored character of all the sort values within this group are non-alphabetical. This type of group occurs when using the alphabetic sort methods listed in table 5.2 or with the letter sort methods listed in table 5.3 or with the letter-number sort methods listed in table 5.4. The alphabetic sort methods ignore many punctuation characters, so an entry that has a non-alphabetic initial character in the sort value may actually be placed in a letter group. The group label is obtained from `\bibglsothergroup`.

- **empty group** The sort value is empty when sorting with an alphabetical, letter or letter-number method, typically a result of the original value consisting solely of commands that `bib2gls` can’t interpret. The group label is obtained from `\bibglsemptygroup`.

- **number group** The entries were sorted by one of the numeric comparisons listed in table 5.5. The group label is obtained from `\bibglnumbergroup`.

- **date-time group** The entries were sorted by one of the date-time comparisons listed in table 5.6 (where both date and time are present). The group label is obtained from `\bibglsdatetimegroup`.

- **date group** The entries were sorted by one of the date comparisons (where the time is omitted). The group label is obtained from `\bibglsdategroup`.

- **time group** The entries were sorted by one of the time comparisons (where the date is omitted). The group label is obtained from `\bibglstimegroup`.

- **custom group** The group label is explicitly set either by aliasing a field (with `field-aliases`) or by using the `group={⟨label⟩}` resource option. You will need to use `\glsxtrsetgrouptitle` in the document to provide an associated title if the `⟨label⟩` isn’t the same as the title. Remember that with older \LaTeX kernels, the label can’t contain any active characters, so you can’t use non-ASCII characters in `⟨label⟩` with `inputenc` (but you
can use non-ASCII alphanumerics with fontspec). To ensure better support for UTF-8 with pdflatex, make sure you have a recent \LaTeX{} distribution and up-to-date versions of glossaries and glossaries-extra.

The letter group titles will typically have the first character converted to upper case for the alphabet sort methods (table 5.2). A “letter” may not necessarily be a single character (depending on the sort rule), but may be composed of multiple characters, such as a digraph (two characters) or trigraph (three characters).

For example, if the sort rule recognises the digraph “dz” as a letter, then it will be converted to “Dz” for the group title. There are some exceptions to this. For example, the Dutch digraph “ij” should be “IJ” rather than “Ij”. This is indicated by the following line in the language resource file:

<entry key="grouptitle.case.ij">IJ</entry>

If there isn’t a grouptitle.case.<lc> key (where <lc> is the lower case version), then only the first character will be converted to upper case otherwise the value supplied by the resource file is used. This resource key is only checked for the alphabetical comparisons listed in table 5.2. If the initial part of the sort value isn’t recognised as a letter according to the sort rule, then the entry will be in a non-letter group (even if the character is alphabetical).

The letter (table 5.3) and letter-number (table 5.4) methods only select the first character of the sort value for the group. If the character is alphabetical\footnote{according to Java’s Character.isAlphabetic(int) method} then it will be a letter group otherwise it’s a non-letter group. The case-insensitive ordering (such as sort={letter-nocase}) will convert the letter group character to upper case. The case-sensitive ordering (such as sort={letter-case}) won’t change the case.

Glossary styles with navigational links to groups (such as indexhypergroup) require an extra run for the ordinary \makeglossaries and \makenoidxglossaries methods. For example, for the document myDoc.tex:

\begin{verbatim}
pdflatex myDoc
makeglossaries myDoc
pdflatex myDoc
pdflatex myDoc
\end{verbatim}

On the first \texttt{pdflatex} call, there’s no glossary. On the second \texttt{pdflatex}, there’s a glossary but the glossary must be processed to find the group information, which is written to the \texttt{.aux} file as

\texttt{@gls@hypergroup{⟨type⟩}{⟨group id⟩}}

The third \texttt{pdflatex} reads this information and is then able to create the navigation links.

With \texttt{bib2gls}, if the type is provided (through the \texttt{type} field or via options such as \texttt{type} and \texttt{dual-type}) then this information can be determined when \texttt{bib2gls} is ready to write the \texttt{.glstex} file, which means that the extra \LaTeX{} run isn’t necessary. If \texttt{bib2gls} doesn’t know the glossary type then it will fallback on the original method which requires an extra \LaTeX{} run.

For example:
3.1 Common Options

Here the type is set and bib2gls can detect that hyperref has been loaded, so if the --group switch is used, then the group hyperlinks can be set (using \bibglshypergroup). This means that the build process is just:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,abbreviations,style={indexhypergroup}]{glossaries-extra}
\GlsXtrLoadResources[src={entries}, % data in entries.bib
type={main}] % put these entries in the 'main' glossary
\GlsXtrLoadResources[src={abbrvs}, % data in abbrvs.bib
type={abbreviations}] % put entries in the 'abbreviations' glossary

Note that this requires glossaries v4.32+. If your version of glossaries is too old then bib2gls can’t override the default behaviour of glossary-hypernav’s \glsnavhypertarget.

If hyperref isn’t loaded or the --group switch isn’t used or the type isn’t set or your version of glossaries is too old, then the information can’t be saved in the .glstex file.

For example:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,abbreviations,style={indexhypergroup}]{glossaries-extra}
\GlsXtrLoadResources[src={entries}] % data in entries.bib
\GlsXtrLoadResources[src={abbrvs}] % data in abbrvs.bib

This requires the build process:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,abbreviations,style={indexhypergroup}]{glossaries-extra}
\GlsXtrLoadResources[src={entries}] % data in entries.bib
\GlsXtrLoadResources[src={abbrvs}] % data in abbrvs.bib

because the group hyperlink information can’t be determined by bib2gls, so it’s best to always set the type if you want hyper-group styles, and make sure you have an up-to-date version of glossaries (and glossaries-extra).
--no-group

Don’t automatically set the group field with group={auto} (default). The glossary won’t have groups even if a group style, such as indexgroup, is used (unless the group field is set to a custom value).

3.2 File Options

--dir ⟨dirname⟩ (or -d ⟨dirname⟩)

By default bib2gls assumes that the output files should be written in the current working directory. The input .bib files are assumed to be either in the current working directory or on \TeX’s path (in which case kpsewhich will be used to find them).

If your .aux file isn’t in the current working directory (for example, you have run \TeX with -output-directory) then you need to take care how you invoke bib2gls.

Suppose I have a file called test-entries.bib that contains my entry definitions and a document called mydoc.tex that selects the .bib file using:

\GlsXtrLoadResources[src={test-entries}]

(test-entries.bib is in the same directory as mydoc.tex). If I compile this document using

pdflatex -output-directory tmp mydoc

then the auxiliary file mydoc.aux will be written to the tmp sub-directory. The resource information is listed in the .aux file as

\glsxtr@resource{src={test-entries}}{mydoc}

If I run bib2gls from the tmp directory, then it won’t be able to find the test-entries.bib file (since it’s in the parent directory).

If I run bib2gls from the same directory as mydoc.tex using

bib2gls tmp/mydoc

then the .aux file is found and the transcript file is tmp/mydoc.glg (since the default path name is the same as the .aux file but with the extension changed to .glg) but the output file mydoc.glstex will be written to the current directory.

This works fine from \TeX’s point of view as it can find the .glstex file, but it may be that you’d rather the .glstex file was tidied away into the tmp directory along with all the other files. In this case you need to invoke bib2gls with the --dir or -d option:

bib2gls -d tmp mydoc
3.3 Interpreter Options

--log-file ⟨filename⟩ (or -t ⟨filename⟩)
Sets the name of the transcript file. By default, the name is the same as the .aux file but with a .glg extension. Note that if you use bib2gls in combination with xindy or makeindex, you will need to change the transcript file name to prevent conflict.

--tex-encoding ⟨name⟩
bib2gls tries to determine the character encoding to use for the output files. If the document has loaded the inputenc package then bib2gls can obtain the value of the encoding from the .aux file. This is then converted to a name recognised by Java. For example, utf8 will be mapped to UTF-8. If the fontspec package has been loaded, glossaries-extra will assume the encoding is utf8 and write that value to the .aux file.

If neither package has been loaded, bib2gls will assume the Java Virtual Machine’s (JVM) default encoding (identified by the file.encoding property). If this is incorrect or if bib2gls can’t work out the appropriate mapping then you can specify the correct encoding using --tex-encoding ⟨name⟩ where ⟨name⟩ is the encoding name (such as UTF-8).

If you have a problem with non-ASCII characters not displaying correctly in your document:

• Check that the file encoding of your document .tex file (or files) has been correctly set by your text editor.

• Check that your document supports that encoding (for example, through the inputenc package).

• Check bib2gls’s transcript file (.glg) for the line that starts

    TeX character encoding:

    This should be followed by the encoding used by bib2gls when creating the .glstex files. If this is incorrect use --tex-encoding.

• Check that the encoding of the .bib files (set by your text editor or bibliographic management system) matches the encoding line in the .bib file or the charset resource option.

3.3 Interpreter Options

--break-space
The interpreter treats a tilde character - as a normal space. Similarly \nobreakspace just produces a space.
3.3 Interpreter Options

--no-break-space
The interpreter treats a tilde character - as a non-breakable space (default). Similarly \nobreakspace produces a non-breakable space character (0x00A0).

--custom-packages \langle list \rangle
Instruct the interpreter to parse the package files identified in \langle list \rangle. The package files need to be quite simple. When this switch is used, the interpreter can recognise \ProvidesPackage, \DeclareOptions (and \DeclareOptions*), \ProcessOptions, \PackageError and \RequirePackage, but it can’t deal with complicated code. In the case of \RequirePackage, support will also be governed by --custom-packages. This option has a cumulative action.

--ignore-packages \langle list \rangle (or -k \langle list \rangle)
This option is cumulative. When the document .log file is parsed for known packages, bib2gls will skip the check for any listed in \langle list \rangle. Note that this option simply instructs bib2gls to ignore the package information in the log file. Any packages that are identified with --packages will be passed to the interpreter if support is available, even if the package is also listed in --ignore-packages. Note that unknown packages can’t be included in the ignored \langle list \rangle.

--interpret
Switch on the interpreter mode (default). See chapter 2 for more details.

--no-interpret
Switch off the interpreter mode. See chapter 2 for more details about the interpreter.

--list-known-packages
This option will list all the packages supported by the \TeX parser library and will then exit bib2gls. The results are divided into two sections: those packages that are searched for in the .log file and those packages that aren’t searched for in the .log file but have some support available. Some of the support is very limited. Package options aren’t detected. The transcript file is always searched for glossaries-extra to ensure that the version is new enough to support bib2gls.

Packages that fall into the first category are: amsmath, amsymb, bpchem, fontenc, fontsymb, fourier, hyperref, lipsum, MnSymbol, mhchem, natbib, pifont, siunitx (limited), stix, textcase, textcomp, tipa, upgreek and wasysym. (You can omit checking for specific packages with --ignore-packages.) These are packages that provide commands that might be needed within entry fields. The check for fontsymb is to simply determine whether or not UTF-8
characters are allowed in labels (for labelify and labelify-list). (Now that there is better support for UTF-8 with pdfLaTeX, UTF-8 characters will be allowed in labels if the detected versions of glossaries and glossaries-extra are new enough, but note that you will also need a relatively new \LaTeXX kernel as well.)

Packages that fall into the second category are: booktabs, color, datatool-base (very limited), datatool (very limited), etoolbox (very limited), graphics, graphicx, ifthen, jmlrutils, mfirstuc-english, probsoln, shortvrb, and xspace. These are less likely to be needed within fields and so aren’t checked for by default. If they are needed then you can instruct bib2gls to support them with --packages.

Note that mfirstuc is always automatically loaded, but mfirstuc-english is not implemented unless explicitly requested with --packages mfirstuc-english.

If you’re wondering about the selection, the texparserlib.jar library was originally written for another application that required support for some of them.

\textbf{--packages \langle list \rangle (or -p \langle list \rangle)}

Instruct the interpreter to assume the packages listed in \langle list \rangle have been used by the document. This option has a cumulative action so --packages "wasysym,pifont" is the same as --packages wasysym --packages pifont.

Note that there’s only a limited number of packages supported by the \TeX parser library. This option is provided for cases where you’re using a command from a package that the interpreter doesn’t support but it happens to have the same name and meaning as a command from a package that the interpreter does support. You can also use it to provide support for known packages that aren’t checked for when the .log file is parsed. If you want bib2gls to parse an unsupported package use --custom-packages.

\textbf{--support-unicode-script}

Text superscript (\textsuperscript) and subscript (\textsubscript) will use Unicode super/subscript characters if available (default). For example,

\texttt{\textsuperscript{(2)}}

will be converted to \textsuperscript{(2)}, which consists of: 0x207D (superscript left parenthesis) 0x00B2 (superscript two) 0x207E (superscript right parenthesis). If the entire contents of the argument can’t be represented by Unicode characters, the interpreter uses <sup> and <sub> markup, which is then stripped by bib2gls. For example,

\texttt{\textsuperscript{(2,3)}}

will be converted to

<sup>(2,3)</sup>

(since there’s no superscript comma). The markup is stripped leaving just (2,3).
3.4 Record Options

Superscripts and subscripts in maths mode always use markup regardless of this setting. Some supported packages that use ^ or _ as shortcuts within an encapsulating command may internally use the same code as \textsuperscript and \textsubscript, in which case they will be sensitive to this setting.

--no-support-unicode-script
Text superscript (\textsuperscript) and subscript (\textsubscript) won’t use Unicode super/subscript characters. Note that if other commands are provided that expand to Unicode superscript or subscript characters, then they won’t be affected by this setting. For example, if \superiortwo is defined as
\providecommand{\superiortwo}{\char"B2}
then it will be interpreted as 0x00B2 (superscript two) even if this setting is on.

--obey-aux-catcode
By default, the .aux parser ignores category code changing commands. This option will instruct the parser to implement the category code, but note that it can only do this for known commands that the parser is able to implement.

--no-obey-aux-catcode
Instructs the .aux parser to ignore category code changing commands. (Default.)

3.4 Record Options

--cite-as-record
Treat instances of \citation{⟨label⟩} found in the .aux file as though it was actually an ignored record:
\glsxtr@record{⟨label⟩}{⟨page⟩}{glsignore}{
Note that \citation{*} will always be skipped. Use selection={all} to select all entries. This switch is most useful in conjunction with @bibtexentry (page 97).

--no-cite-as-record
Don’t check for instances of \citation in the .aux file (default).

--collapse-same-location-range
Collapse any explicit range into a normal record if the start and end locations are the same (default). This record will be treated as a normal location that can be merged with neighbouring locations, regardless of merge-ranges.
### 3.4 Record Options

**--no-collapse-same-location-range**

Don’t collapse any explicit range into a normal record if the start and end locations are the same. The explicit range will only be able to merge with neighbouring locations if `merge-ranges={true}`.

**--map-format** *(map:value list)* *(or* `-m (map:value list)`*)*

This sets up the rule of precedence for partial location matches (see section 5.10). The argument may be a comma-separated list of *(map):(value)* pairs. Alternatively, you can have multiple instances of `--map-format (map):(value)` which have a cumulative effect.

For example,

```bash
text2gls --map-format "emph:hyperbf" mydoc
```

This essentially means that if there’s a record conflict involving `emph`, try replacing `emph` with `hyperbf` and see if that resolves the conflict.

Note that if the conflict includes a range formation, the range takes precedence. The mapping tests are applied as the records are read. For example, suppose the records are listed in the `.aux` file as:

```latex
\glsxtr@record{gls.sample}{}{page}{emph}{3}
\glsxtr@record{gls.sample}{}{page}{hypersf}{3}
\glsxtr@record{gls.sample}{}{page}{hyperbf}{3}
```

and **text2gls** is invoked with

```bash
text2gls --map-format "emph:hyperbf,hypersf:hyperit" mydoc
```

or

```bash
text2gls --map-format emph:hyperbf --map-format hypersf:hyperit mydoc
```

then **text2gls** will process these records as follows:

1. Accept the first record *(emph)* since there’s currently no conflict. (This is the first record for page 3 for the entry given by `gls.sample`.)

2. The second record *(hypersf)* conflicts with the existing record *(emph)*. Neither has the format `glsnumberformat` or `glsignore` so **text2gls** consults the mappings provided by `--map-format`.
   - The `hypersf` format (from the new record) is mapped to `hyperit`, so **text2gls** checks if the existing record has this format. In this case it doesn’t (the format is `emph`). So **text2gls** moves on to the next test:
3.4 Record Options

- The `emph` format (from the existing record) is mapped to `hyperbf`, so `bib2gls` checks if the new record has this format. In this case it doesn’t (the format is `hypersf`).

  Since the provided mappings haven’t resolved this conflict, the new record is discarded with a warning. Note that there’s no look ahead to the next record. (There may be other records for other entries also used on page 3 interspersed between these records.)

3. The third record (`hyperbf`) conflicts with the existing record (`emph`). Neither has the format `glsnumberformat` or `glsignore` so `bib2gls` again consults the mappings provided by `--map-format`.

- The new record’s `hyperbf` format has no mapping provided, so `bib2gls` moves on to the next test:
  - The existing record’s `emph` format has a mapping provided (`hyperbf`). This matches the new record’s format, so the new record takes precedence.

  This means that the location list ends up with the `hyperbf` location for page 3.

If, on the other hand, the mappings are given as

```bash
--map-format "emph:hyperit,hypersf:hyperit,hyperbf:hyperit"
```

then all the three conflicting records (`emph`, `hypersf` and `hyperbf`) will end up being replaced by a single record with `hyperit` as the format.

Multiple conflicts will typically be rare as there’s usually little reason for more than two or three different location formats within the same list. (For example, `glsnumberformat` as the default and `hyperbf` or `hyperit` for a principal location.)

`--merge-nameref-on ⟨rule⟩`

The `record={nameref}` package option (introduced to glossaries-extra version 1.37) provides extra information in the record when indexing, obtained from `\@currentlabelname`, `\@currentHref` and `\theHentrycounter`. Instead of writing the record as:

```latex
\glsxtr@record{⟨label⟩}{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}
```

the record is written as:

```latex
\glsxtr@record@nameref{⟨label⟩}{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨title⟩}{⟨href⟩}{⟨hcounter⟩}
```

If `hyperref` hasn’t been loaded `⟨title⟩` and `⟨href⟩` will always be empty. The most reliable target is given by `⟨counter⟩.⟨hcounter⟩`, where `⟨counter⟩` is the associated counter name and `⟨hcounter⟩` is obtained from `\theHentrycounter`, which is set to the hyper target command `\theH{⟨counter⟩}` during indexing. Since this information can’t be included in the location when indexing with `makeindex` or `xindy`, the base `glossaries` package tries to obtain a prefix
from which the target name can be formed. This doesn’t work if the \H\langle\texttt{counter}\rangle can’t be formed from \langle\texttt{prefix}\rangle the\langle\texttt{counter}\rangle, which results in broken links. Since bib2gls doesn’t have the same restrictions, the actual target can be included in the record. You can then customize the document to choose whether to use \langle\texttt{href}\rangle (to link to the nearest anchor) or \langle\texttt{hcounter}\rangle to link to the place where the indexing counter was incremented.

The \texttt{nameref} record will be written to the location list using:

\begin{verbatim}
glsxtrdisplaylocnameref{\langle\texttt{prefix}\rangle}{\langle\texttt{counter}\rangle}{\langle\texttt{format}\rangle}{\langle\texttt{location}\rangle}{\langle\texttt{title}\rangle}{\langle\texttt{href}\rangle}{\langle\texttt{hcounter}\rangle}{\langle\texttt{file}\rangle}
\end{verbatim}

The \langle\texttt{file}\rangle part will be empty for normal internal locations, and will be set to the corresponding file name for supplemental locations.

With hyperref, \langle\texttt{title}\rangle is initially empty. The \langle\texttt{href}\rangle will be Doc-Start at the start of the document and is updated globally on every instance of \refstepcounter. The \langle\texttt{title}\rangle is updated locally by certain commands, such as \texttt{section} or \texttt{caption}. This means that the \langle\texttt{href}\rangle may not always correspond to the \langle\texttt{title}\rangle, so using the record=\texttt{nameref} package option can have unpredictable results if the \langle\texttt{title}\rangle is used as link text with \langle\texttt{href}\rangle as the target.

For compactness, bib2gls tries to merge duplicate or near duplicate records. There are four possible rules that it will use for \texttt{nameref} records, identified by \langle\texttt{rule}\rangle in the --merge-nameref-on switch:

- \texttt{location}: merge records that match on the \langle\texttt{prefix}\rangle, \langle\texttt{counter}\rangle and \langle\texttt{location}\rangle parts (as regular records);
- \texttt{title}: merge records that match on the \langle\texttt{counter}\rangle and \langle\texttt{title}\rangle parts;
- \texttt{href}: merge records that match on the \langle\texttt{counter}\rangle and \langle\texttt{href}\rangle parts;
- \texttt{hcounter}: merge records that match on the \langle\texttt{counter}\rangle and \langle\texttt{hcounter}\rangle parts.

The default \langle\texttt{rule}\rangle is \texttt{hcounter}. Note that for all rules the \langle\texttt{counter}\rangle must match. See the “Nameref Record” section of the glossaries-extra user manual for further details.

\texttt{--merge-wrglossary-records}

For use with the indexcounter package option (glossaries-extra v1.29+), this switch merges an entry’s wrglossary records for the same page location. This is the default setting. (See also \texttt{save-index-counter}.)

\texttt{--no-merge-wrglossary-records}

Don’t merge an entry’s wrglossary records. This means that you may end up with duplicate page numbers in the entry’s location list, but they will link to different parts of the page.
--record-count (or -c)

Switch on record counting. This will ensure that when each entry is written to the .glstex file, bib2gls will additionally set the following fields:

- `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.

These fields can then be used with the \rgls-like commands.

This option is governed by the --record-count-rule, which can be used to exclude certain types of records from the count. The default rule is all, which includes all ignored records.

The default behaviour of

\rgls[(options)]{(label)}[(insert)]

is to check the `recordcount` field against the `recordcount` attribute value. This attribute can be set with

\GlsXtrSetRecordCountAttribute{⟨category list⟩}{⟨value⟩}

where `⟨category list⟩` is a comma-separated list of category labels and `⟨value⟩` is a positive integer. If the value of the `recordcount` field is greater than `⟨value⟩` then \rgls behaves like \gls, otherwise it does

\rglsformat{(label)}[(insert)]

instead. If the use of \rglsformat is triggered in this way, then \rgls writes a record to the .aux file with the `format` set to `glstriggerrecordformat`. This ensures that the record count is correct on the next run, but the record isn’t added to the location list as bib2gls recognises it as a special ignored record. Note that the entry will still appear in the usual glossary unless you assign it to a different one with `trigger-type`.

If the `recordcount` attribute hasn’t been set \rgls behaves like \gls. (That is, \rgls uses the same internal command used by \gls.) You can use \glsxtrablenablerecordcount to redefine \gls to \rgls, so that you can continue to use \gls without having to switch command name.

For example:

\GlsXtrLoadResources[
  src={abbrevs},'% entries defined in abbrevs.bib
  trigger-type={ignored},
  category={abbreviation}
]
\glsxtrablenablerecordcount
\GlsXtrSetRecordCountAttribute{abbreviation}{1}
3.4 Record Options

See the glossaries-extra user manual [13] for further details.

Take care not to confuse the recordcount field with the indexed field. The indexed field keeps a running total of the number of times an entry has been recorded so far, and is updated every time the entry is indexed during the current \LaTeX run. The recordcount field stores the total number of records obtained by bib2gls from the .aux file.

\texttt{--no-record-count}

Switch off record counting. (Default.)

\texttt{--record-count-unit} (or \texttt{-n})

Automatically implements \texttt{--record-count} and additionally sets the recordcount.\{counter\}.\{location\} fields. These fields can then be used with the \texttt{\textbackslash gls}-like commands. This option is governed by \texttt{--record-count-rule}, to determine which records should be counted.

\texttt{--no-record-count-unit}

Switches off unit record counting. (Default.) Note that you need \texttt{--no-record-count} to completely switch off record counting.

\texttt{--record-count-rule} \{rule\} (or \texttt{-r} \{rule\})

Automatically implements \texttt{--record-count} and sets the rule that determines which records should contribute to the count. The \{rule\} may be one of:

- all or a: these keywords indicate that all records should be included in the count (default).
- non-ignored or n: these keywords indicate that ignored records should be excluded in the count.
- c/\{regex\}/: only records where the associated counter name matches the regular expression \{regex\} should be included in the count.
- f/\{regex\}/: only records where the associated format matches the regular expression \{regex\} should be included in the count.
- f/\{format-regex\}/c/\{counter-regex\}/\langle op\rangle: this combines the format and counter name match. The trailing \{op\} is optional. If present, it should be one of the keywords: and (boolean AND) or or (boolean OR). If omitted, and is assumed.

For example:
3.5 Bib File Options

```
bib2gls --record-count-rule 'f/.*(bf|it)/c/(sub)?section/or' myDoc
```

This will only count records where the format matches the regular expression .*(bf|it) (for example, hyperbf or hyperit) or the counter name matches section or subsection (but not subsection, since the expressions are anchored).

This syntax doesn’t permit the use of the sequence /c/ appearing in the regular expressions, but both the format and counter name are either control sequence names or are a substring of a control sequence name, so they should typically just be alphabetical strings.

```
--retain-formats ⟨list⟩
```

It’s possible that you may not want to lose certain location formats, even if it means having duplicate locations. For example, if you want to move a principal location using `save-principal-locations={remove}`. In which case, use this switch with a comma-separated list of formats that should be retained. Note that exact duplicates will still be merged. This switch has a cumulative effect.

Take care if you use this switch and you have an explicit range with coincident start and end locations. If the principal record is between the start and end format markers then the range can’t collapse to an ordinary record. (You may need to use `merge-ranges={true}.`)

```
--no-retain-formats
```

Normal location merging rules apply (default).

3.5 Bib File Options

```
--warn-non-bib-fields
```

If any internal fields are found in the `.bib` file, this setting will issue a warning as their use can cause unexpected results. The fields checked for are those listed in Tables 4.5 and 4.6 with a few exceptions, notably `type` and `sort`. Ideally you shouldn’t need to use `sort` as there should be an appropriate fallback set up to use if `sort` isn’t set, such as the label for symbols or the name for terms or the short form for abbreviations (see section 5.8).

This is the default setting and was added as some users were confused over which fields could be used in the `.bib` file. The use of these fields can break `bib2gls`’s normal behaviour and cause unexpected results.

The check is performed before field aliasing, so it’s possible to alias a field to an internal field, such as `group`, without triggering this warning. If you do this you need to make sure you have taken appropriate precautions to avoid unexpected results.

```
--no-warn-non-bib-fields
```

Switches off the check for non-bib fields. If you use this option you need to make sure you have taken appropriate precautions to avoid unexpected results.
--warn-unknown-entry-types

If any unknown entry types are found in the .bib file, bib2gls will issue a warning with this option set (default).

--no-warn-unknown-entry-types

This option will suppress the warning if an unknown entry types are found in the .bib file.

3.6 Field Options

--no-expand-fields

By default, \newglossaryentry and similar commands expand field values (except for name, symbol and description). This is useful if constructing field values programatically (for example in a loop) but can cause a problem if certain fragile commands are included in the field.

The switch --no-expand-fields makes bib2gls write "\glsnoexpandfields" to the .glstex file, which switches off the expansion. Since bib2gls is simply fetching the data from .bib files, it’s unlikely that this automatic expansion is required and since it can also be problematic this option is on by default. You can switch it off with --expand-fields.

--expand-fields

Don’t write "\glsnoexpandfields" to the .glstex file, allowing fields to expand when the entries are defined. Remember that this doesn’t include the name, symbol or description fields, which need to have their expansion switched on with \glssetexpandfield before the entries are defined (that is, before using \GlsXtrLoadResources).

--mfirstuc-protection ⟨list⟩|all (or -u ⟨list⟩|all)

If you have mfirstuc v2.08+, glossaries v4.50+ and glossaries-extra v1.49+ then this setting shouldn’t be required any more as there’s now better sentence-case handling. If these versions are detected in the .log file then the default will switch to --no-mfirstuc-protection otherwise the default is --mfirstuc-protection. If this causes any problems, use --mfirstuc-protection to re-enable this setting. The information below relates to older versions.

Commands like \Gls use \makefirstuc provided by the mfirstuc package. This command has limitations and one of the things that can break it is the use of a referencing command at the start of its argument. The glossaries-extra package has more detail about the problem in the “Nested Links” section of the user manual [13]. If a glossary field starts with one of these problematic commands, the recommended method (if the command can’t be replaced) is to insert an empty group in front of it.

For example, the following definition
\newabbreviation{shtml}{shtml}{\glsps{ssi} enabled \glsps{short}{html}}

will cause a problem for \gls{shtml} on first use. The above example would be written in a .bib file as:

@abbreviation{shtml,
  short={shtml},
  long={\glsps{ssi} enabled \glsps{html}}
}

The default \texttt{mfirstuc} protection will automatically insert an empty group before \texttt{\gls{ssi}} when writing the definition in the .glistex file.

The argument for this switch should either be a comma-separated list of fields or the keyword all (which indicates all fields). \texttt{bib2gls} will automatically insert an empty group at the start of the listed fields that start with a problematic command, and a warning will be written to the transcript. Unknown fields are skipped even if they’re included in the list. An empty argument is equivalent to \texttt{--no-mfirstuc-protection}. The default value is all.

\texttt{--no-mfirstuc-protection}

Switches off the \texttt{mfirstuc} protection mechanism described above.

\texttt{--mfirstuc-math-protection}

If you have \texttt{mfirstuc v2.08+}, \texttt{glossaries v4.50+} and \texttt{glossaries-extra v1.49+} then this setting shouldn’t be required any more as there’s now better sentence-case handling. If these versions are detected in the .log file then the default will switch to \texttt{--no-mfirstuc-math-protection}. If this causes any problems, use \texttt{--mfirstuc-math-protection} to re-enable this setting. The information below relates to older versions.

This setting works in the same way as \texttt{--mfirstuc-protection} but guards against fields starting with inline maths ($...$). For example, if the \texttt{name} field starts with $x$ and the glossary style automatically tries to convert the first letter of the name to upper case, then this will cause a problem.

With \texttt{--mfirstuc-math-protection} set, \texttt{bib2gls} will automatically insert an empty group at the start of the field and write a warning in the transcript. This setting is on by default.

\texttt{--no-mfirstuc-math-protection}

Switches off the above.

\texttt{--nested-link-check \langle list\rangle|none}

By default, \texttt{bib2gls} will parse certain fields for potential nested links. (See the section “Nested Links” in the glossaries-extra user manual [13].)
The default set of fields to check are: name, text, plural, first, firstplural, long, longplural, short, shortplural and symbol.

You can change this set of fields using --nested-link-check \langle value \rangle where \langle value \rangle may be none (don’t parse any of the fields) or a comma-separated list of fields to be checked.

--no-nested-link-check

Equivalent to --nested-link-check none.

--shortcuts \langle value \rangle

Some entries may reference another entry within a field, using commands like \gls, so bib2gls parses the fields for these commands to determine dependent entries to allow them to be selected even if they haven’t been used within the document. The shortcuts package option provided by glossaries-extra defines various synonyms, such as \ac which is equivalent to \gls. By default the value of the shortcuts option will be picked up by bib2gls when parsing the .aux file. This then allows bib2gls to additionally search for those shortcut commands while parsing the fields.

You can override the shortcuts setting using --shortcuts \langle value \rangle (where \langle value \rangle may take any of the allowed values for the shortcuts package option), but in general there is little need to use this switch.

--trim-fields

Trim leading and trailing spaces from all field values. For example, if the .bib file contains:

```plaintext
@entry{sample,
    name = {sample},
    description = {
        an example
    }
}
```

This will cause spurious spaces in the description field. Using --trim-fields will automatically trim the values before writing the .glistex file.

Note that even without this trimming option on, fields that are set as keys within \longnewglossaryentry or the optional argument of \newabbreviation will automatically have the leading and trailing spaces internally trimmed by the xkeyval package, so this trimming action only affects fields that aren’t set in this way, such as the description, long and short fields. If you specifically require a space at the start or end of a field then use a spacing command, such as \␣ or \space or ~.
3.7 Other Options

--trim-only-fields \langle list \rangle

Only trim leading and trailing spaces from the fields identified in the comma-separated \langle list \rangle. This option has a cumulative effect but is cancelled by --no-trim-fields (which switches off all trimming) and by --trim-fields (which switches on trimming for all fields). This option may not be used with --trim-except-fields.

For example, to only trim the description field:

bib2gls --trim-only-fields description myDoc

--trim-except-fields \langle list \rangle

Trim all leading and trailing spaces from fields except those identified in the comma-separated \langle list \rangle. This option has a cumulative effect but is cancelled by --no-trim-fields (which switches off all trimming) and by --trim-fields (which switches on trimming for all fields). This option may not be used with --trim-only-fields. See the above note about xkeyval.

For example, to trim all fields except short and long:

bib2gls --trim-except-fields short,long myDoc

Or

bib2gls --trim-except-fields short --trim-except-fields long myDoc

--no-trim-fields

Don’t trim any leading or trailing spaces from field values (but see the above note about xkeyval). This is the default setting.

3.7 Other Options

--force-cross-resource-refs (or -x)

Force cross-resource reference mode on (see section 1.4).

--no-force-cross-resource-refs

Don’t force cross-resource reference mode on (default). The mode will be enabled if applicable (see section 1.4).

--provide-glossaries

This setting will make bib2gls add the line

\provideignoredglossary*{\langle type \rangle}
3.7 Other Options

to the .glstex file before an entry is defined where that entry has the type field set to an unknown glossary type (bib2gls can detect from the .aux file all glossaries that have been defined with \newglossary but not those defined with \newignoredglossary).

This ensures that the glossary exists, but the use of \provideignoredglossary (rather than \newignoredglossary) will prevent an error if the glossary has already been defined.

--no-provide-glossaries

This setting prevents bib2gls from providing unknown glossaries, except in a few documented situations (the master, trigger-type and secondary options). This is the default since it’s a useful way of detecting misspelt glossary labels. It’s harder to detect the problem if a misspelt label has caused an entry to be added to a hidden glossary.

--replace-quotes

Single and double-quote characters (’ and ”) will be written as \bibglsaposchar and \bibglsdoublequotechar in field values and group information written to the .glstex file.

--no-replace-quotes

Single and double-quote characters (’ and ”) will be written as those actual characters (default).
4 .bib Format

bib2gls recognises certain entry types. Any unrecognised types will be ignored and a warning will be written to the transcript file. Entries are defined in the usual .bib format:

\{@⟨entry-type⟩\{(id)\},
   ⟨field-name-1⟩ = {⟨text⟩},
   ...
   ⟨field-name-n⟩ = {⟨text⟩}
\}

where ⟨entry-type⟩ is the entry type (listed below), ⟨field-name-1⟩, ..., ⟨field-name-n⟩ are the field names and ⟨id⟩ is a unique label. The label can’t contain any spaces or commas, and most special characters are forbidden. The hyphen character and some other punctuation characters are allowed by bib2gls, but you need to make sure that your document hasn’t made them active. In general it’s best to stick with alpha-numeric labels. The field values may be delimited by braces {⟨text⟩} or double-quotes "⟨text⟩".

The label-prefix option can be used to instruct bib2gls to insert prefixes to the labels ((id)) when the data is read. Remember to use these prefixes when you reference the entries in the document, but don’t include them when you reference them in the .bib file. There are some special prefixes that have a particular meaning to bib2gls: “dual.” and “ext⟨n⟩.” where ⟨n⟩ is a positive integer. In the first case, dual . references the dual element of a dual entry (see @dualentry). This prefix will be replaced by the value of the dual-prefix option. The ext⟨n⟩. prefix is used to reference an entry from a different set of resources (loaded by another \GlsXtrLoadResources command). This prefix is replaced by the corresponding element of the list supplied by ext-prefixes, but this is only supported if the cross-resource reference mode is enabled (see section 1.4).

In the event that the sort value falls back on the label, the original label supplied in the .bib file is used, not the prefixed label.

4.1 Encoding

Avoid non-ASCII characters in the ⟨id⟩ if your document uses the inputenc package. (This isn’t a problem for XƎLƎTEX or LuaXƎTEX, but you still need to avoid special characters.) You can set the character encoding in the .bib file using:

% Encoding: ⟨encoding-name⟩

where ⟨encoding-name⟩ is the name of the character encoding. For example:
4.2 Comments

% Encoding: UTF-8

You can also set the encoding using the `charset` option, but it’s simpler to include the above comment on the first line of the `.bib` file. (This comment is also searched for by JabRef to determine the encoding, so it works for both applications.) If you don’t use either method `bib2gls` will have to search the entire `.bib` file, which is inefficient and you may end up with a mismatched encoding.

Note that recent changes to the \LaTeX kernel now allow non-ASCII characters in labels when using commands such as `\label` (with `inputenc`). The commands used by the `glossaries` package are more complicated, but changes have been made in `glossaries v4.47` and `glossaries-extra v1.46` to help support this, however it hasn’t been fully tested.

4.2 Comments

The original `.bib` file format as defined by \LaTeX doesn’t have a designated comment character, but instead treats anything outside of `@entry{entry}{data}` as unwanted material that’s ignored. This can catch out users who try to do something like:

```latex
%@misc{sample, title={Sample} }
```

In this case, the percent character is simply discarded and the line is treated as:

```latex
@misc{sample, title={Sample} }
```

Some applications that parse `.bib` files are less tolerant of unwanted material. In the case of `bib2gls`, the percent character is treated as a comment character and other unwanted material should be omitted. Avoid using comments within field values. Comments are best placed outside of entry definitions.

The most common type of comment is the encoding comment, described above. \LaTeX’s `@comment` is also supported by `bib2gls` for general comments, but not for the encoding.

4.3 Fields

Each entry type may have required fields, optional fields and ignored fields. These are set using a key=value list within `@entry-type{entry}{id}{fields}` in the `.bib` file. Most keys recognised by \newglossaryentry may be used as a field unless `bib2gls` considers them an internal field (see below). In general, you shouldn’t need to use the `sort` field.

If an optional field is missing and `bib2gls` needs to access it for some reason, `bib2gls` will try to fallback on another value. The actual fallback value depends on the entry type. The most common fallback is that used if the `sort` field is missing, which is typically the case. This approach allows different entry types to have different fields used for sorting (see section 5.8).

Predefined fields for use in `.bib` files are listed in Tables 4.1, 4.2, 4.3 and 4.4. If you add any custom keys in your document using \glsaddkey or \glsaddstoragekey, those commands must be placed before the first use of \GlsXtrLoadResources to ensure that `bib2gls` recognises them as a valid field name.
If you define your own custom keys, ensure that they don’t contain spaces, commas (,), equal signs (=) or any other character that isn’t supported by the .bib format. Additionally, if you want to use assign-fields, ensure that you don’t use any of the assignment special characters, such as plus (+), within any field names.

Internal fields that may be assigned within the document (the \LaTeX assignment code having been written by bib2gls in the .glstex file) are listed in Table 4.5. These typically shouldn’t be used in the .bib file. Some of these fields can be set for a particular document using a resource option, such as type or group. With --warn-non-bib-fields set, bib2gls will check for internal fields that can cause interference with its normal operations and will warn if any are found in the .bib file.

There are also some fields that are set and used by glossaries or glossaries-extra listed in Table 4.6 that aren’t recognised by bib2gls. In most cases these fields don’t have a designated key and are only intended for internal use by bib2gls or by the glossaries or glossaries-extra package. Note that the value of the sort field written to the .bib file doesn’t always exactly match the sort value used by bib2gls (which is stored in bib2gls@sort). Any special characters found in the sort value are always substituted before writing the .bib file to avoid syntax errors.

Any unrecognised fields will be ignored by bib2gls. This is more convenient than using \input or \loadglsentries, which requires all the keys used in the file to be defined, regardless of whether or not you actually need them in the document.

Other entries can be cross-referenced using the see, seealso or alias fields or by using commands like \gls or \glsxtrp in any of the recognised fields. These will automatically be selected if the selection setting includes dependencies, but you may need to rebuild the document to ensure the location lists are correct. Use of the \glssee command will create an ignored record and the see field will be set to the relevant information. If an entry has the see field already set, any instance of \glssee in the document for that entry will be appended to the see field (provided you have at least v1.14 of glossaries-extra). In general, it’s best just to use the see field and not use \glssee.

The seealso key was only added to glossaries-extra v1.16, but this field may be used with bib2gls even if you only have version 1.14 or 1.15. If the key isn’t available, seealso=\{(xr-list)\} will be treated as see=\{[seealso\name]xr-list\} (the resource option seealso won’t have an effect). You can’t use both see and seealso for the same entry with bib2gls. Note that the seealso field doesn’t allow for the optional [\{tag\}] part. If you need a different tag, either use see or change the definition of see\name or \glsxtruseseealsoformat. Note that, unless you are using xindy, \glsxtrindexseealso just does \glssee[seealso\name], and so will be treated as see rather than seealso by bib2gls. Again, it’s better to just use the seealso field directly.

You can identify an arbitrary field as containing a list of dependent entry labels with dependency-fields. This instructs bib2gls to parse the listed fields for dependencies in a similar manner to the see field, but it doesn’t add any information to the cross-referencing part of the location list. The option may be used in combination with the see or seealso fields.
### 4.3 Fields

Table 4.1: Fields Provided by glossaries-extra

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>The entry with this field set is a synonym of the entry whose label is given by this field.</td>
</tr>
<tr>
<td>category</td>
<td>The entry’s category label.</td>
</tr>
<tr>
<td>description</td>
<td>The description displayed in the glossary.</td>
</tr>
<tr>
<td>descriptionplural</td>
<td>The plural form of the description.</td>
</tr>
<tr>
<td>first</td>
<td>The text to display on first use with \gls{⟨label⟩}.</td>
</tr>
<tr>
<td>firstplural</td>
<td>The text to display on first use with \glspl{⟨label⟩}.</td>
</tr>
<tr>
<td>long</td>
<td>The long form of an abbreviation. (Set internally by commands like \newabbreviation.)</td>
</tr>
<tr>
<td>longplural</td>
<td>The plural long form of an abbreviation.</td>
</tr>
<tr>
<td>name</td>
<td>The name displayed in the glossary.</td>
</tr>
<tr>
<td>nonumberlist</td>
<td>Used to suppress the location list for a specific entry. Its value may only be true or false. Technically this isn’t actually a field as its value isn’t saved so it can’t be referenced or modified after the entry has been defined.</td>
</tr>
<tr>
<td>parent</td>
<td>The parent entry’s label. See section 1.2.</td>
</tr>
<tr>
<td>plural</td>
<td>The text to display on subsequent use of \glspl{⟨label⟩}.</td>
</tr>
<tr>
<td>see</td>
<td>General purpose cross-reference (syntax: see={⟨tag⟩⟨xr-list⟩}).</td>
</tr>
<tr>
<td>seealso</td>
<td>Cross-reference related entries (syntax: seealso={⟨xr-list⟩}).</td>
</tr>
<tr>
<td>short</td>
<td>The short form of an abbreviation. (Set internally by commands like \newabbreviation.)</td>
</tr>
<tr>
<td>shortplural</td>
<td>The plural short form of an abbreviation.</td>
</tr>
<tr>
<td>symbol</td>
<td>The associated symbol.</td>
</tr>
<tr>
<td>symbolplural</td>
<td>The plural form of the associated symbol.</td>
</tr>
<tr>
<td>text</td>
<td>The text to display on subsequent use of \gls{⟨label⟩}.</td>
</tr>
<tr>
<td>user1</td>
<td>A general purpose user field.</td>
</tr>
<tr>
<td>user2</td>
<td>A general purpose user field.</td>
</tr>
<tr>
<td>user3</td>
<td>A general purpose user field.</td>
</tr>
<tr>
<td>user4</td>
<td>A general purpose user field.</td>
</tr>
<tr>
<td>user5</td>
<td>A general purpose user field.</td>
</tr>
<tr>
<td>user6</td>
<td>A general purpose user field.</td>
</tr>
</tbody>
</table>
## 4.3 Fields

### Table 4.2: Fields Provided by bib2gls

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adoptparents</td>
<td>The list of adopted parents for entries spawned by @progenitor. (Field only available for use in .bib file within @progenitor-like entries.)</td>
</tr>
<tr>
<td>dualdescription</td>
<td>May be used to identify a dual description.</td>
</tr>
<tr>
<td>duallong</td>
<td>The long form of a dual abbreviation mapped by @dualabbreviation.</td>
</tr>
<tr>
<td>duallongplural</td>
<td>The plural long form of a dual abbreviation mapped by @dualabbreviation.</td>
</tr>
<tr>
<td>dualprefix</td>
<td>The dual of the prefix field. This field isn’t provided with a key or associated command, but can be accessed as an internal field.</td>
</tr>
<tr>
<td>dualprefixfirst</td>
<td>The dual of the prefixfirst field. This field isn’t provided with a key or associated command, but can be accessed as an internal field.</td>
</tr>
<tr>
<td>dualprefixfirstplural</td>
<td>The dual of the prefixfirstplural field. This field isn’t provided with a key or associated command, but can be accessed as an internal field.</td>
</tr>
<tr>
<td>dualprefixplural</td>
<td>The dual of the prefixplural field. This field isn’t provided with a key or associated command, but can be accessed as an internal field.</td>
</tr>
<tr>
<td>dualshort</td>
<td>The short form of a dual abbreviation mapped by @dualabbreviation.</td>
</tr>
<tr>
<td>dualshortplural</td>
<td>The plural short form of a dual abbreviation mapped by @dualabbreviation.</td>
</tr>
</tbody>
</table>

### Table 4.3: Fields Provided by glossaries-prefix

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>The prefix associated with the text field.</td>
</tr>
<tr>
<td>prefixfirst</td>
<td>The prefix associated with the first field.</td>
</tr>
<tr>
<td>prefixfirstplural</td>
<td>The prefix associated with the firstplural field.</td>
</tr>
<tr>
<td>prefixplural</td>
<td>The prefix associated with the plural field.</td>
</tr>
</tbody>
</table>

### Table 4.4: Fields Provided by glossaries-accsupp

Don’t load glossaries-accsupp directly (with \usepackage) when using glossaries-extra. Load using the accsupp package option instead.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>The replacement text for the name field.</td>
</tr>
</tbody>
</table>
4.3 Fields

Fields Provided by glossaries-accsupp (Continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>descriptionaccess</td>
<td>The replacement text for the description field.</td>
</tr>
<tr>
<td>descriptionpluralaccess</td>
<td>The replacement text for the descriptionplural field.</td>
</tr>
<tr>
<td>firstaccess</td>
<td>The replacement text for the first field.</td>
</tr>
<tr>
<td>firstpluralaccess</td>
<td>The replacement text for the firstplural field.</td>
</tr>
<tr>
<td>longaccess</td>
<td>The replacement text for the long field.</td>
</tr>
<tr>
<td>longpluralaccess</td>
<td>The replacement text for the longplural field.</td>
</tr>
<tr>
<td>pluralaccess</td>
<td>The replacement text for the plural field.</td>
</tr>
<tr>
<td>shortaccess</td>
<td>The replacement text for the short field.</td>
</tr>
<tr>
<td>shortpluralaccess</td>
<td>The replacement text for the shortplural field.</td>
</tr>
<tr>
<td>symbolaccess</td>
<td>The replacement text for the symbol field.</td>
</tr>
<tr>
<td>symbolpluralaccess</td>
<td>The replacement text for the symbolplural field.</td>
</tr>
<tr>
<td>textaccess</td>
<td>The replacement text for the text field.</td>
</tr>
</tbody>
</table>

Table 4.5: Fields Sometimes Set by bib2gls in the .glistex File

You may define and assign bibtextype as a key (although it’s more likely to be aliased). Don’t define any of the others listed in this table, and don’t use any of them in the .bib file. A possible exception is the type field, but it’s more flexible to set that through a resource option. The explicit use of group within a .bib file can cause unpredictable results and is best set through a resource option or by bib2gls. In general, you shouldn’t need to set the sort field as appropriate fallbacks should produce useful sort values (see section 5.8).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bibtexcontributor</td>
<td>An internal list field provided when a @contributor entry is automatically created by @bibtexentry.</td>
</tr>
<tr>
<td>bibtexentry</td>
<td>An internal list field created by @bibtexentry.</td>
</tr>
<tr>
<td>bibtexentry⟨entry-type⟩</td>
<td>An internal list field created by @bibtexentry.</td>
</tr>
<tr>
<td>bibtextype</td>
<td>Used by bib2gls as a substitution for BibTEX’s type field when parsing @bibtexentry. Needs to be defined or aliased to make it available in the document.</td>
</tr>
<tr>
<td>childcount</td>
<td>Stores the number of children this entry has had selected.</td>
</tr>
<tr>
<td>childlist</td>
<td>A list of labels (in etoolbox’s internal list format) of the children this entry has had selected.</td>
</tr>
<tr>
<td>counter</td>
<td>The default counter used for indexing (assigned by the counter option).</td>
</tr>
<tr>
<td>definitionindex</td>
<td>Stores the definition index.</td>
</tr>
<tr>
<td>dual</td>
<td>Created by dual-field if set with no value, this field is used to store the dual label.</td>
</tr>
</tbody>
</table>
### 4.3 Fields

Fields Sometimes Set by bib2gls in the .glistex File (Continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨field⟩endpunc</td>
<td>Used with the check-end-punctuation option.</td>
</tr>
<tr>
<td>group</td>
<td>The letter group determined by the comparator (or assigned by the group option). See section 1.2.</td>
</tr>
<tr>
<td>indexcounter</td>
<td>Stores the location corresponding to the matching wrglossary reference.</td>
</tr>
<tr>
<td>location</td>
<td>The typeset location list.</td>
</tr>
<tr>
<td>loclist</td>
<td>The internal list of locations.</td>
</tr>
<tr>
<td>originalentrytype</td>
<td>The original entry type before any aliasing was applied or the actual entry type if no aliasing.</td>
</tr>
<tr>
<td>originalid</td>
<td>The original label as given in the .bib file.</td>
</tr>
<tr>
<td>primarylocations</td>
<td>Stores the locations that use one of the designated primary formats, if enabled.</td>
</tr>
<tr>
<td>progenitor</td>
<td>The label identifying the @progenitor that spawned this entry.</td>
</tr>
<tr>
<td>progeny</td>
<td>A comma-separated list of labels identifying the entries spawned by @progenitor.</td>
</tr>
<tr>
<td>recordcount</td>
<td>Used with record counting to store the total record count.</td>
</tr>
<tr>
<td>recordcount.⟨counter⟩</td>
<td>Used with record counting to store the total number of records for a given counter.</td>
</tr>
<tr>
<td>recordcount.⟨counter⟩.⟨location⟩</td>
<td>Used with record counting to store the total number of records for a given location.</td>
</tr>
<tr>
<td>rootancestor</td>
<td>Stores the label of this entry’s root ancestor.</td>
</tr>
<tr>
<td>secondarygroup</td>
<td>The letter group determined by the comparator used with the secondary sort.</td>
</tr>
<tr>
<td>secondarysort</td>
<td>The sort value determined by the comparator used with the secondary sort.</td>
</tr>
<tr>
<td>siblingcount</td>
<td>Stores the number of siblings this entry has had selected.</td>
</tr>
<tr>
<td>siblinglist</td>
<td>A list of labels (in etoolbox’s internal list format) of the siblings this entry has had selected.</td>
</tr>
<tr>
<td>sort</td>
<td>The sort value obtained by the comparator.</td>
</tr>
<tr>
<td>type</td>
<td>The glossary this entry belongs to (assigned by the type option). See section 1.2.</td>
</tr>
<tr>
<td>useindex</td>
<td>Stores the order of use index.</td>
</tr>
</tbody>
</table>
4.3 Fields

Table 4.6: Internal Fields Set by glossaries or glossaries-extra or bib2gls

Don’t define any of these as keys and don’t use any of them in the .bib file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bib2gls@sort</td>
<td>Used by bib2gls to store the actual sort value.</td>
</tr>
<tr>
<td>bib2gls@sortfallback</td>
<td>Used by bib2gls to store the sort fallback value.</td>
</tr>
<tr>
<td>currcount</td>
<td>Used with entry counting to store the current total.</td>
</tr>
<tr>
<td>currcount@⟨value⟩</td>
<td>Used with unit entry counting (glossaries-extra).</td>
</tr>
<tr>
<td>desc</td>
<td>Corresponds to description key.</td>
</tr>
<tr>
<td>descplural</td>
<td>Corresponds to descriptionplural key.</td>
</tr>
<tr>
<td>firstpl</td>
<td>Corresponds to firstplural key.</td>
</tr>
<tr>
<td>flag</td>
<td>Boolean that determines if an entry has been used.</td>
</tr>
<tr>
<td>index</td>
<td>The main part of the indexing code (makeindex or xindy).</td>
</tr>
<tr>
<td>indexed</td>
<td>The value is incremented everytime the entry is indexed.</td>
</tr>
<tr>
<td>level</td>
<td>Hierarchical level.</td>
</tr>
<tr>
<td>longpl</td>
<td>Corresponds to longplural key.</td>
</tr>
<tr>
<td>prenumberlist</td>
<td>set by the nonumberlist entry key with \makenoidxglossaries</td>
</tr>
<tr>
<td>prevcount</td>
<td>Used with entry counting to store the total from the previous run.</td>
</tr>
<tr>
<td>prevcount@⟨value⟩</td>
<td>Used with unit entry counting (glossaries-extra).</td>
</tr>
<tr>
<td>prevunitmax</td>
<td>Used with unit entry counting (glossaries-extra).</td>
</tr>
<tr>
<td>prevunittotal</td>
<td>Used with unit entry counting (glossaries-extra).</td>
</tr>
<tr>
<td>shortpl</td>
<td>Corresponds to shortplural key.</td>
</tr>
<tr>
<td>sortvalue</td>
<td>Original sort value (before sanitizing and escaping special characters).</td>
</tr>
<tr>
<td>unitlist</td>
<td>Used with unit entry counting (glossaries-extra).</td>
</tr>
<tr>
<td>useri</td>
<td>Corresponds to user1 key.</td>
</tr>
<tr>
<td>userii</td>
<td>Corresponds to user2 key.</td>
</tr>
<tr>
<td>useriii</td>
<td>Corresponds to user3 key.</td>
</tr>
<tr>
<td>useriv</td>
<td>Corresponds to user4 key.</td>
</tr>
<tr>
<td>userv</td>
<td>Corresponds to user5 key.</td>
</tr>
<tr>
<td>uservi</td>
<td>Corresponds to user6 key.</td>
</tr>
</tbody>
</table>

Table 4.7: Compound Set Fields

Only available for @compoundset. These correspond to the arguments of \multiglossary-entry.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements</td>
<td>Only available for @compoundset this required field should contain a comma-separated list of labels.</td>
</tr>
</tbody>
</table>
### 4.3 Fields

Compound Set Fields (Continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Only available for <code>@compoundset</code> this optional field should contain the main label. If omitted, the final element from the <code>elements</code> field is assumed.</td>
</tr>
<tr>
<td>option</td>
<td>Only available for <code>@compoundset</code> this optional field should contain the default options that govern the set (which override conflicting options set with <code>\multiglossaryentrysetup</code> and can be overridden by options to commands like <code>\mgls</code>).</td>
</tr>
</tbody>
</table>
4.4 String Concatenation

The .bib format allows you to perform string concatenation. That is, join fragments together to form a single value. The concatenation operator in .bib files is #. For example, if the following string is defined:

@string{markuplang={markup language}}

Then values can be obtained by concatenating this string with other strings. For example:

@abbreviation{xml,
  short={XML},
  long={extensible } # markuplang
}
@abbreviation{html,
  short={HTML},
  long={hypertext } # markuplang
}

This is equivalent to:

@abbreviation{xml,
  short={XML},
  long={extensible markup language}
}
@abbreviation{html,
  short={HTML},
  long={hypertext markup language}
}

Note that some resource options allow string concatenation in their syntax. That uses a different operator. See section 5.1 for further details.

4.5 Standard Entry Types

@string

The standard @string is available and can be used to define variables that may be used in field values. Don’t include braces or double-quote delimiters when referencing a variable. You can use # to concatenate strings. For example:

@string{ssi={server-side includes}}
@string{html={hypertext markup language}}

@abbreviation{shtml,
  short="shtml",

4.5 Standard Entry Types

```latex
\longtt=ssi # " enabled " # html,
\see={ssi,html}
\}
\@abbreviation\{html,
\short=html",
\long=html
\}
\@abbreviation\{ssi,
\short=ssi",
\long=ssi
\}
```

Note the difference between "ssi" (a field value delimited by double-quotes), the undelimited =ssi (a reference to the variable), the grouped ={ssi,html} (a field value delimited by braces) and ssi the entry label.

\@preamble

The standard \@preamble is available and can be used to provide command definitions used within field values. For example:

```latex
\@preamble{"\providecommand{\mtx}[1]{\boldsymbol{#1}}"}
```

```latex
\@entry\{matrix,
\name=\{matrix\},
\plural=\{matrices\},
\description=\{rectangular array of values, denoted $\mtx{M}$\}
\}
```

Alternatively you can use \texttt{\glsxtrprovidecommand} which behaves the same as \texttt{\providecommand} within the document but behaves like \texttt{\renewcommand} within \texttt{bib2gls}, which allows you to change \texttt{bib2gls}'s internal definition of a command without affecting the definition within the document (if it's already been defined before the resource file is input). In general, it's best to just use \texttt{\providecommand}.

The \LaTeX{} parser library used by \texttt{bib2gls} will parse the contents of \@preamble before trying to interpret the field value used as a fallback when \texttt{sort} is omitted (unless \texttt{interpret-preamble=\{false\}} is set in the resource options). For example:

```latex
\@preamble{"\providecommand{\set}[1]{\mathcal{#1}}\providecommand{\card}[1]{|\set{#1}|}"
```

```latex
\@entry\{S,
\name=\{\$\set{S}\$\},
```

60
4.5 Standard Entry Types

Neither entry has the `sort` field, so `bib2gls` has to fall back on the `name` field and, since this contains the special characters \ (backslash), $ (maths shift), \{ (begin group) and \} (end group), the \TeX parser library is used to interpret it. The definitions provided by \texttt{@preamble} allow bib2gls to deduce that the `sort` value of the $S$ entry is just $S$ and the `sort` value of the \texttt{card} entry is $|S|$ (see chapter 2).

What happens if you also need to use these commands in the document? The definitions provided in \texttt{@preamble} won’t be available until the .\texttt{glstex} file has been created, which means the commands won’t be defined on the first \LaTeX run.

There are several approaches:

1. Just define the commands in the document. This means the commands are available, but bib2gls won’t be able to correctly interpret the `name` fields.

2. Define the commands in both the document and in \texttt{@preamble}. For example:

   \begin{verbatim}
   \newcommand{\set}[1]{\mathcal{#1}}
   \newcommand{\card}[1]{|\set{#1}|}
   \GlsXtrLoadResources[src={my-data}]
   \end{verbatim}

   Alternatively:

   \begin{verbatim}
   \GlsXtrLoadResources[src={my-data}]
   \providecommand{\set}[1]{\mathcal{#1}}
   \providecommand{\card}[1]{|\set{#1}|}
   \end{verbatim}

   If the provided definitions match those given in the .\texttt{bib} file, there’s no difference. If they don’t match then in the first example the document definitions will take precedence (but the interpreter will use the \texttt{@preamble} definitions) and in the second example the \texttt{@preamble} definitions will take precedence. For example, the document may define \texttt{\card} as:

   \begin{verbatim}
   \newcommand{\card}[1]{\vert\set{#1}\vert}
   \end{verbatim}

3. Make use of \texttt{\glsxtrfmt} provided by \texttt{glossaries-extra} which allows you to store the name of the formatting command in a field. The default is the \texttt{user1} field, but this can be changed to another field by redefining \texttt{\GlsXtrFmtField}.

   The .\texttt{bib} file can now look like this:
4.5 Standard Entry Types

@preamble{"\providecommand{\set}[1]{\mathcal{#1}}
\providecommand{\card}[1]{|\set{#1}|}"
}

@symbol{S,
        name={{}$\set{S}$},
        text={\set{S}},
        user1={set},
        description={a set}
}
@symbol{cardS,
        name={{}$\card{S}$},
        text={\card{S}},
        user1={card},
        description={the cardinality of \gls{S}}
}

Within the document, you can format \text{⟨text⟩} using the formatting command provided in the \text{user1} field with:

\glsxtrfmt[⟨options⟩]{⟨label⟩}{⟨text⟩}

(which internally uses \glslink) or

\glsxtrentryfmt{⟨label⟩}{⟨text⟩}

which just applies the appropriate formatting command to \text{⟨text⟩}. Version 1.23+ of glossaries-extra also provides a starred form of the linking command:

\glsxtrfmt*[⟨options⟩]{⟨label⟩}{⟨text⟩}{⟨insert⟩}

which inserts additional material inside the link text but outside the formatting command.

If the entry given by \text{⟨label⟩} hasn’t been defined, then \glsxtrfmt just does \text{⟨text⟩} (followed by \text{⟨insert⟩} for the starred version) and a warning is issued. (There’s no warning if the entry is defined but the field hasn’t been set.) The \text{⟨options⟩} are as for \glslink but \glslink will actually be using:

\glslink[⟨def-options⟩,⟨options⟩]{⟨label⟩}{\langle\text{text}⟩}{⟨csname⟩}{⟨insert⟩}

where the default options \text{⟨def-options⟩} are given by \text{\GlsXtrFmtDefaultOptions}. The default definition of this is just noindex which suppresses the automatic indexing or recording action. (See the glossaries-extra manual [13] for further details.) The \text{⟨insert⟩} part is omitted for the unstarred form.

This means that the document doesn’t need to actually provide \text{\set} or \text{\card} but can instead use, for example,
4.5 Standard Entry Types

\glsxtrfmt{S}\{A\}
\glsxtrentryfmt{cardS}\{B\}

instead of:
\set{A}
\card{B}

The first \LaTeX run will simply ignore the formatting and produce a warning. Since this is a bit cumbersome to write, you can provide shortcut commands. For example:
\GlsXtrLoadResources[src={my-data}]
\newcommand{\gset}[2]{\glsxtrfmt[#1]{S}{#2}}
\newcommand{\gcard}[2]{\glsxtrfmt[#1]{cardS}{#2}}

Whilst this doesn’t seem a great deal different from simply providing the definitions of \set and \card in the document, this means you don’t have to worry about remembering the names of the actual commands provided in the \texttt{.bib} file (just the entry labels) and the use of \glsxtrfmt will automatically produce a hyperlink to the glossary entry if the \hyperref\ package has been loaded.

Here’s an alternative \texttt{.bib} that defines entries with a term, a description and a symbol:

\begin{verbatim}
@preamble{"\providecommand{\setfmt}[1]{\mathcal{#1}}
\providecommand{\cardfmt}[1]{|\setfmt{#1}|}"}

@entry{set,
  name={set},
  symbol={\setfmt{S}},
  user1={setfmt},
  description={collection of values}
}
@entry{cardinality,
  name={cardinality},
  symbol={\cardfmt{S}},
  user1={cardfmt},
  description={the number of elements in the \gls{set} \glssymbol{set}}
}
\end{verbatim}

I’ve changed the entry labels and the names of the formatting commands. The definitions in the document need to reflect the change in label but not the change in the formatting commands:
Here’s another approach that allows for a more complicated argument for the cardinality. (For example, if the argument is an expression involving set unions or intersections.) The .bib file is now:

\begin{verbatim}
@preamble{"
providecommand{\setfmt}[1]{\mathcal{#1}}
providecommand{\cardfmt}[1]{|#1|}"
}

@entry{set,
  name={set},
  symbol={\setfmt{S}},
  user1={setfmt},
  description={collection of values}
}
@entry{cardinality,
  name={cardinality},
  symbol={\cardfmt{\setfmt{S}}},
  user1={cardfmt},
  description={the number of elements in the \glss{set} $\glssymbol{set}$} 
}
\end{verbatim}

This has removed the \setfmt command from the definition of \cardfmt. Now the definitions in the document are:

\begin{verbatim}
\newcommand{\gset}[2][{}\glssxtrfmt[#1]{set}{#2}]
\newcommand{\gcard}[2][{}\glssxtrfmt[#1]{cardinality}{#2}]
\end{verbatim}

This allows for code such as:

\begin{verbatim}
[ \gcard{\gset{A} \cap \gset{B}} \ ]
\end{verbatim}

which will link back to the cardinality entry in the glossary and avoids any hyperlinking with \gset. Alternatively to avoid links with \gcard as well:

\begin{verbatim}
\newcommand{\gset}[1][{}\glssxtrentryfmt{set}{#1}]
\newcommand{\gcard}[1][{}\glssxtrentryfmt{cardinality}{#1}]
\end{verbatim}

Now \gset and \gcard are simply formatting commands, but their actual definitions are determined in the .bib file.

4.6 Single Entry Types

The entry types described in this section create a single glossary definition per entry (from glossaries-extra’s point of view). For example:
4.6 Single Entry Types

The *secondary* option allows the creation of a fake glossary with the entry labels in its internal list in a different order. This means that the same data can be displayed in two separate lists without duplicating the resources required by each glossary entry.

Section 4.7 describes *bib2gls* entry types that create two separate (but related) glossaries-extra definitions per .bib entry.

@entry

Regular terms are defined by the @entry field. This requires the *description* field and either *name* or *parent*. For example:

@preamble{"\providecommand{\mtx}[1]{\boldsymbol{#1}}"}

@entry{matrix,
    name={matrix},
    plural={matrices},
    description={rectangular array of values, denoted \gls{M}},
    seealso={vector}
}

@entry{M,
    name={\ensuremath{M}},
    description={a \gls{matrix}}
}

@entry{vector,
    name = "vector",
    description = {column or row of values, denoted \gls{v}},
    seealso={matrix}
}
4.6 Single Entry Types

@entry{v,
    name={\ensuremath{\vec{v}}},
    description={a \gls{vector}}
}

If the sort field is missing the default is obtained from the name field (unless overridden by options like entry-sort-fallback). For hierarchical entries, if the name field is omitted it will be obtained from the parent’s name. See section 5.8.

Terms defined using @entry will be written to the output (.glstex) file using the command \bibglsnewentry.

@symbol

The @symbol entry type is much like @entry, but it’s designed specifically for symbols, so in the previous example, the M and v terms would be better defined using the @symbol entry type instead. For example:

@symbol{M,
    name={\ensuremath{M}},
    description={a \gls{matrix}}
}

The required fields are name or parent. The description field is required if the name field is missing. If the sort field is omitted, the default fallback is given by the entry label (unless overridden by options like symbol-sort-fallback). Note that this is different from @entry where the sort defaults to name if omitted. See section 5.8.

Terms that are defined using @symbol will be written to the output file using the command \bibglsnewsymbol.

@number

The @number entry type is like @symbol, but it’s for numbers. The numbers don’t have to be explicit digits and may have a symbolic representation. There’s no real difference between the behaviour of @number and @symbol except that terms defined using @number will be written to the output file using the command \bibglsnewnumber.

For example, the file constants.bib might define mathematical constants like this:

@number{pi,
    name={\ensuremath{\pi}},
    description={the ratio of the length of the circumference of a circle to its diameter},
    user1={3.14159}
}
4.6 Single Entry Types

@number{e, 
    name={\ensuremath{e}},
    description={base of natural logarithms},
    user1={2.71828}
}

This stores the approximate value in the \texttt{user1} field. This can be used to sort the entries in numerical order according to the values rather than the symbols:

\GlsXtrLoadResources[ 
    src={constants},% constants.bib 
    category={number},% set the category for all selected entries 
    sort={double},% numerical double-precision sort 
    sort-field={user1}% sort according to 'user1' field 
]

The \texttt{category={number}} option makes it easy to adjust the glossary format to include the \texttt{user1} field:

\glsdefpostdesc{number}{% 
    \ifglsHasField{user1}{\glsCurrentEntryLabel}{ (approximate value: \glsCurrentFieldValue)}{% 
        {}% 
    }% 
}

@index

The \texttt{@index} entry type is designed for entries that don’t have a description. Only the label is required. If \texttt{name} is omitted, it’s assumed to be the same as the label, even if \texttt{parent} is present. (Note this is different to the fallback behaviour of \texttt{@entry}, which fetches the name from the parent entry.) If the name contains any characters that can’t be used in the label, you must use the \texttt{name} field. If the \texttt{sort} field is missing the default fallback is obtained from the \texttt{name}. Note that the \texttt{@index} entry type is \textit{not} governed by \texttt{entry-sort-fallback} (but it is governed by \texttt{custom-sort-fallbacks}). This allows \texttt{@index} and \texttt{@entry} to have different fallbacks if the \texttt{sort} field is missing. See section 5.8.

Example:

@index{duck} 
@index{goose,plural={geese}} 
@index{sealion,name={sea lion}} 
@index{facade,name={fa\c{c}ade}}

Terms that are defined using \texttt{@index} will be written to the output file using the command \verb|ibglsnewindex|. 

67
@indexplural

The @indexplural entry type is similar to the @index entry type except that the name field, if missing, is obtained from the plural field. If the plural field is missing it’s obtained from the text field with the plural suffix appended. If the text field is missing, it’s obtained from the original entry label. If the sort field is missing the default is obtained from the name field. (As with @index, @indexplural is not governed by entry-sort-fallback, but it is governed by custom-sort-fallbacks.) See section 5.8. All fields are optional. For example:

@indexplural{goose,
    plural = {geese}
}

@indexplural{duck}

@indexplural{chateau,
    text = {ch\textsuperscript{\textasciitilde}ateau},
    plural = {ch\textsuperscript{\textasciitilde}ateaux}
}

This is equivalent to:

@indexplural{goose,
    name = {geese},
    text = {goose},
    plural = {geese}
}

@indexplural{duck,
    name = {ducks},
    text = {duck},
    plural = {ducks}
}

@indexplural{chateau,
    name = {ch\textsuperscript{\textasciitilde}ateaux},
    text = ch\textsuperscript{\textasciitilde}ateau,
    plural = ch\textsuperscript{\textasciitilde}ateaux
}

Terms that are defined using @indexplural will be written to the output file using the command \bibglsnewindexplural.
@abbreviation

The @abbreviation entry type is designed for abbreviations. The required fields are short and long. If the sort key is missing, bib2gls will use the field given by abbreviation-sort-fallback, which defaults to the short field. (If you want an equivalent of \new-dualentry, use @dualabbreviationentry instead.)

If you use sort-field={name} (rather than the default sort-field={sort}), then the fallback for the name field is always the short field, regardless of the abbreviation-sort-fallback setting, unless you use abbreviation-name-fallback to change the fallback for the name field. See section 5.8.

Note that you must set the abbreviation style before loading the resource file to ensure that the abbreviations are defined correctly, however bib2gls has no knowledge of the abbreviation style so it doesn’t know if the description field must be included or if the default sort value isn’t simply the value of the short field.

You can instruct bib2gls to sort by the long field instead using abbreviation-sort-fallback={long}. You can also tell bib2gls to ignore certain fields using ignore-fields, so you can include a description field in the .bib file if you sometimes need it, and then instruct bib2gls to ignore it when you don’t want it.

For example:

```plaintext
@abbreviation{html,
  short = {html},
  long = {hypertext markup language},
  description = {a markup language for creating web pages}
}
```

If you want the long-noshort-desc style, then you can put the following in your document (where the .bib file is called entries-abbrv.bib):

```plaintext
\setabbreviationstyle{long-noshort-desc}
\GlsXtrLoadResources[src={entries-abbrv},
  abbreviation-sort-fallback={long}]
```

Whereas, if you want the long-short-sc style, then you can instead do:

```plaintext
\setabbreviationstyle{long-short-sc}
\GlsXtrLoadResources[src={entries-abbrv},ignore-fields={description}]
```

or to convert the short value to upper case and use the long-short-sm style instead:

```plaintext
\setabbreviationstyle{long-short-sm}
\GlsXtrLoadResources[src={entries-abbrv},
  short-case-change={uc},% convert short value to upper case
  ignore-fields={description}]
```

Case-changing can be applied with short-case-change to convert the case of the short field, as illustrated above. If you use a style that obtains the description from the long
4.6 Single Entry Types

form, but you want to apply a case-change to the description field with description-case-change, then you can copy the long field to the description with replicate-fields={long=description}.

For example, if entries-abbrv.bib contains:

```plaintext
@abbreviation{html,
  short = {html},
  long  = {hypertext markup language}
}
```

then the document may include:

```plaintext
\setabbreviationstyle{long-short-sc}
\GlsXtrLoadResources[src={entries-abbrv},
  description-case-change={firstuc},
  replicate-fields={long=description}]
```

Note that this can cause a problem for styles that set the description field to the long form encapsulated by a style command (such as with the long-em-short-em style) as this will override the style setting.

Similarly, if you want to change the case of the name field:

```plaintext
\setabbreviationstyle{long-short-sc}
\GlsXtrLoadResources[src={entries-abbrv},
  description-case-change={firstuc},
  name-case-change={uc},
  replicate-fields={long=description,short=name}]
```

Again, this will lose any custom formatting command that would usually be applied by the abbreviation style to the name field (and description, if applicable).

Terms defined using @abbreviation will be written to the output file using the command \bibglsnewabbreviation.

@acronym

The @acronym entry type is like @abbreviation except that the term is written to the output file using the command \bibglsnewacronym.

@contributor

The @contributor entry type is primarily provided for use by the @bibtexentry type. You may use it explicitly if you want, but you need to take care that it doesn’t clash with @bibtexentry. It behaves much like @index except that the term is written to the .gls file. There are no required fields. As with @index, if the name field is missing, the fallback value is the entry’s label (see section 5.8). When this entry type is automatically created by @bibtexentry, the name is set to
4.7 Dual Entry Types

\bibglgscontributor{⟨forenames⟩}{⟨von⟩}{⟨surname⟩}{⟨suffix⟩}

If you do explicitly use @contributor you need to make sure it’s defined before the first instance of @bibtexentry that tries to access it, but within the same resource set. If you ensure that the label of @contributor matches the contributor label generated by @bibtexentry then they can have their dependency lists updated, and the bibtexentry and bibtexentry@⟨entry-type⟩ internal fields can be set for the @contributor entry. For example:

@contributor{KnuthDonaldE,
  name={\bibglgscontributor{Donald E.}{}{Knuth}{}},
  description={Famous mathematician and computer scientist who created \TeX}
}

@book{texbook,
  title = {The \TeX book},
  author = {Donald E. Knuth},
  publisher = {Addison-Wesley},
  year = 1986
}

The resource options then need to include:

entry-type-aliases={\GlsXtrBibTeXEntryAliases},
labelify-replace={
  {{ \string\-\string\.}}{}
}

If the @contributor entry is deferred until after the corresponding @bibtexentry then you will end up with a label clash.

4.7 Dual Entry Types

The entry types described in this section create two separate (but related) glossaries-extra entry definitions per .bib entry. The first of these entries is considered the primary entry, and the second is the dual entry. The naming scheme is @dual⟨entry-type⟩ where both the primary and dual are considered to have the same type of entry (such as @dualsymbol where both the primary and dual are functionally like @symbol) or @dual⟨primary⟩⟨dual⟩ where the primary is functionally like @⟨primary⟩ and the dual is functionally like @⟨dual⟩.

If you need a field to store the dual description in (and you’re not simply swapping known fields around), then you can use the special dualdescription field and add it to your map.

If the fields provided by the glossaries-prefix are defined, there will be additional mappings for the special internal fields dualprefix, dualprefixfirst, dualprefixplural, and dualprefixfirstplural.

For example:
4.7 Dual Entry Types

@dualabbreviationentry{svm,
    short = {SVM},
    long = {support vector machine},
    description = {statistical pattern recognition technique}
}

is like:

@abbreviation{svm,
    short = {SVM},
    long = {support vector machine},
}

@entry{dual.svm,
    text = {SVM},
    name = {support vector machine},
    description = {statistical pattern recognition technique}
}

and is analogous to:

\newabbreviation{svm}{SVM}{support vector machine}
\newglossaryentry{dual.svm}{name={support vector machine},text={SVM},
    description={statistical pattern recognition technique}}

but both entries are considered dependent on each other. This means that if you only reference the primary entry (using \gls etc) then the dual entry will still be selected if the selection setting includes dependencies.

The creation of the dual entry involves mapping or copying fields from the primary entry. Each dual entry type has a set of mappings. If a field in the set of mappings is missing, its fallback value is used (see section 5.8). Any fields that aren’t listed in the mappings are simply copied, except for the alias field, which will never be copied to the dual entry, nor can it be mapped. The alias will only apply to the primary entry. The dual entry is given the label ⟨prefix⟩⟨id⟩ where ⟨prefix⟩ is set by the dual-prefix option and ⟨id⟩ is the label supplied in the .bib file.

If dual-sort={combine} then the dual entries will be sorted along with the primary entries, otherwise the dual-sort indicates how to sort the dual entries and the dual entries will be appended to the end of the .glistex file. The dual-sort-field determines what field to use for the sort value if the dual entries should be sorted separately.

Take care if you have a mixture of entry types (such as @dualindexentry, @dualindex-symbol and @index) and you’re not using the default dual-sort={combine}. Remember that the primary entries are all sorted together along with the single entries types described in section 4.7 (but they may be assigned to different glossary types), and then the dual entries are sorted together (but may be assigned to different glossary types). This may result in an odd ordering if some of the primaries and some of the duals are assigned to the same glossary. For example, don’t mix @dualindexabbreviation (duals are abbreviations) with
4.7 Dual Entry Types

@dualabbreviationentry (primaries are abbreviations) when you aren’t using dual-sort = {combine} (unless you have two different glossaries for the primary vs dual abbreviations).

Remember that bib2gls is designed to take advantage of \printunsrtglossary, which simply iterates over all defined entries in the order in which they were defined (or, more precisely, the order of the internal list of entry labels associated with that glossary). The aim of bib2gls is to write the entry definitions to the .glistex file so that the internal list of labels is in the appropriate order.

For example, suppose the file entries.bib contains:

@index{aardvark}
@index{mouse}
@index{zebra}
@dualindexabbreviation{xml,
  short={XML},
  long={extensible markup language}
}
@dualabbreviationentry{ssi,
  short={SSI},
  long={server-side includes},
  description={directives placed in \gls{html} pages evaluated by the server}
}
@dualindexabbreviation{html,
  short={HTML},
  long={hypertext markup language}
}
@dualabbreviationentry{css,
  short={CSS},
  long={cascading stylesheets},
  description={a language that describes the style of an \gls{html} document}
}

This contains a mixture of entry types, including @dualindexabbreviation (where the dual is the abbreviation) and @dualabbreviationentry (where the primary is the abbreviation).

Now consider the following document:

\documentclass{article}
\usepackage[record,abbreviations]{glossaries-extra}
\GlsXtrLoadResources[selection={all},src={entries}]
\begin{document}
\printunsrtglossaries

73
4.7 Dual Entry Types

This uses the default sort={combine}, so all the entries are sorted together, resulting in the order: aardvark, dual.css, css, html, dual.html, mouse, dual.ssi, ssi, xml, dual.xml, zebra.

The \LaTeX code written to the .glsstex file is essentially (but not exactly):

\begin{verbatim}
\% from @index{aardvark}:
newglossaryentry{aardvark}{name={aardvark},description={}}

\% dual of @dualabbreviationentry{css,...}:
newglossaryentry{dual.css}{name={cascading stylesheets},\{text\}={CSS},
description={a language that describes the style of an
\glsxtrshort{html} document}}

\% primary of @dualabbreviationentry{css,...}:
newabbreviation{css}{CSS}{cascading stylesheets}

\% primary of @dualindexabbreviation{html,...}:
newglossaryentry{html}{name={HTML},description={}}

\% dual of @dualindexabbreviation{html,...}:
newabbreviation{dual.html}{HTML}{hypertext markup language}

\% from @index{mouse}:
newglossaryentry{mouse}{name}={mouse},description={}

\% dual of @dualabbreviationentry{ssi,...}:
newglossaryentry{dual.ssi}{name={server-side includes},\{text\}={SSI},
description={directives placed in \glsxtrshort{html} pages
evaluated by the server}}

\% primary of @dualabbreviationentry{ssi,...}:
newabbreviation{ssi}{SSI}{server-side includes}

\% primary of @dualindexabbreviation{xml,...}:
newglossaryentry{xml}{name={XML},description={}}

\% dual of @dualindexabbreviation{xml,...}:
newabbreviation{dual.xml}{XML}{extensible markup language}

\% from @index{zebra}:
newglossaryentry{zebra}{name={zebra},description={}}
\end{verbatim}

Since the document uses the abbreviations package option, \newabbreviation automatically assigns the abbreviation to the abbreviations glossary (created through that package
option). This means that the main (default) glossary contains the entries (in order):

- aardvark (name: aardvark),
- dual.css (name: cascading stylesheets),
- html (name: HTML),
- mouse (name: mouse),
- dual.ssi (name: server-side includes),
- xml (name: XML),
- zebra (name: zebra).

The abbreviations glossary contains:

- css (short: CSS),
- dual.html (short: HTML),
- ssi (short: SSI),

Since all the entries were combined and sorted together, the resulting glossaries are both ordered alphabetically (using short for the abbreviations and name for the rest), but note that you need to take care when referencing the abbreviations if you want to make use of the abbreviation style. You need \gls{css} and \gls{ssi} for the primary abbreviations created with \dualabbreviationentry and \gls{dual.html} and \gls{dual.xml} for the dual abbreviations created with \dualindexabbreviation. Also the name of the primary/dual alternative of the abbreviations is also inconsistent (short form for html and xml and long form for dual.css and dual.ssi), as different field mappings are used.

If the document is changed so that the dual entries are now sorted and written after all the primary entries have been dealt with:

\GlsXtrLoadResources[  
src={entries},  
dual-sort=letter-nocase,  
selection=all  
]

then \bib2gls first orders the primaries:

- aardvark (name: aardvark),
- css (short: CSS),
- html (name: HTML),
4.7 Dual Entry Types

- `mouse` (name: mouse),
- `ssi` (short: SSI),
- `xml` (name: XML),
- `zebra` (name: zebra)

and writes them to the `.glistex` file (functionally like):

```latex
% from @index{aardvark}:
\newglossaryentry{aardvark}{name={aardvark},description={}}

% primary of @dualabbreviationentry{css,…}:
\newabbreviation{css}{CSS}{cascading stylesheets}

% primary of @dualindexabbreviation{html,…}:
\newglossaryentry{html}{name={HTML},description={}}

% from @index{mouse}:
\newglossaryentry{mouse}{name={mouse},description={}}

% primary of @dualabbreviationentry{ssi,…}:
\newabbreviation{ssi}{SSI}{server-side includes}

% primary of @dualindexabbreviation{xml,…}:
\newglossaryentry{xml}{name={XML},description={}}

% from @index{zebra}:
\newglossaryentry{zebra}{name={zebra},description={}}

Then `bib2gls` orders the duals:

- `dual.css` (name: cascading stylesheets),
- `dual.html` (short: HTML),
- `dual.ssi` (name: server-side includes),
- `dual.xml` (short: XML)

and writes them to the `.glistex` file (functionally like):

```latex
% dual of @dualabbreviationentry{css,…}:
\newglossaryentry{dual.css}{name={cascading stylesheets},text={CSS},
description={a language that describes the style of an\glsxtrshort{html} document}}
```

76
When the `.glstex` file is input (during the next \LaTeX{} run) the entries are defined in the order:

1. aardvark (type: main),
2. css (type: abbreviations),
3. html (type: main),
4. mouse (type: main),
5. ssi (type: abbreviations),
6. xml (type: main),
7. zebra (type: main),
8. dual.css (type: main),
9. dual.html (type: abbreviations),
10. dual.ssi (type: main),
11. dual.xml (type: abbreviations).

This means that the main glossary’s internal list is in the order:

- aardvark (aardvark),
- html (HTML),
- mouse (mouse),
- xml (XML),
- zebra (zebra),
- dual.css (cascading stylesheets),
4.7 Dual Entry Types

- dual.ssi (server-side includes)

and the abbreviations glossary's internal list is in the order:

- css (CSS),
- ssi (SSI),
- dual.html (HTML),
- dual.xml (XML).

The lists are no longer in alphabetical order as they have a mixture of primary and dual entries that were separated before sorting.

The above is a fairly contrived example as it wouldn’t make sense in a real document to have glossary terms (that include a description) mixed with index terms (that don’t include a description). A better solution would be to use @tertiaryindexabbreviationentry instead of @dualabbreviationentry.

@dualentry

The @dualentry entry type is similar to @entry but actually defines two entries. The dual entry contains the same information as the primary entry but some of the fields are swapped around. The default mappings are:

- name \(\mapsto\) description
- plural \(\mapsto\) descriptionplural
- description \(\mapsto\) name
- descriptionplural \(\mapsto\) plural

If the prefix fields are defined, then the default mappings additionally include:

- prefix \(\mapsto\) dualprefix
- prefixplural \(\mapsto\) dualprefixplural
- prefixfirst \(\mapsto\) dualprefixfirst
- prefixfirstplural \(\mapsto\) dualprefixfirstplural
- dualprefix \(\mapsto\) prefix
- dualprefixplural \(\mapsto\) prefixplural
- dualprefixfirst \(\mapsto\) prefixfirst
- dualprefixfirstplural \(\mapsto\) prefixfirstplural
The required fields are as for @entry.

For example:

@dualentry{child,  
    name={child},  
    plural={children},  
    description={enfant} 
} 

is like:

@entry{child,  
    name={child},  
    plural={children},  
    description={enfant}  
    descriptionplural={enfants} 
} 

@entry{dual.child,  
    description={child},  
    descriptionplural={children},  
    name={enfant}  
    plural={enfants} 
} 

where dual. is replaced by the value of the dual-prefix option. However, instead of defining the entries with \bibglsnewentry both the primary and dual entries are defined using \bibglsnewdualentry. The category and type fields can be set for the dual entry using the dual-category and dual-type options.

For example:

\newglossary*{english}{English}  
\newglossary*{french}{French}  

\GlsXtrLoadResources[  
    src={entries-dual},%! data in entries-dual.bib  
    type={english},%! put primary entries in glossary 'english'  
    dual-type={french},%! put dual entries in glossary 'french'  
    category={dictionary},%! set the primary category to 'dictionary'  
    dual-category={dictionary},%! set the dual category to 'dictionary'  
    sort={en},%! sort primary entries according to language 'en'  
    dual-sort={fr}%! sort dual entries according to language 'fr' 
]  

If you need to keep the same name but have different descriptions then you can use dual-description and set up a mapping to use it. For example:
4.7 Dual Entry Types

@dualentry{sample,
  name={sample},
  description={primary sample description},
  dualdescription={dual sample description}
}

The mapping can then be:

dual-entry-map={{description},
  {dualdescription}}

@dualindexentry

There are no required fields. The primary entry behaves like @index and the dual entry behaves like @entry. The default field mapping is:

- name → name

If the prefix fields are defined, then the default mappings additionally include:

- prefix → dualprefix
- prefixplural → dualprefixplural
- prefixfirst → dualprefixfirst
- prefixfirstplural → dualprefixfirstplural
- dualprefix → prefix
- dualprefixplural → prefixplural
- dualprefixfirst → prefixfirst
- dualprefixfirstplural → prefixfirstplural

This doesn't actually perform any swapping of fields, but it provides the field used for back-links (if dual-indexentry-backlink is set). The reason that the primary (rather than the dual) is like @index is to allow the primaries to merge with any @index entries found in the resource set, since glossary entries with descriptions are likely to be a subset of all indexed entries.

If no name is given, the dual entry is assigned the (unprefixed) entry label. For example:

@dualindexentry{array,
  description={ordered list of values}
}

This is effectively like:
The primary entries are defined using \bibglsnewdualindexentry, which by default sets the category to index (although this may be overridden, for example, by the category option). The dual entries are defined with \bibglsnewdualindexentrysecondary.

This is the most convenient way of having an entry that’s also automatically indexed. For example, suppose the file terms.bib contains:

@index{duck}
@index{zebra}
@index{aardvark}

and suppose the file entries.bib contains:

@dualindexentry{array,
   description={ordered list of values}
}
@dualindexentry{vector,
   name={vector},
   description={column or row of values}
}
@dualindexentry{set,
   description={collection of values}
}
@dualindexentry{matrix,
   plural={matrices},
   description={rectangular array of values}
}

These entries can be used in an example document that has an index and a glossary:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,index,stylemods={mcols}]{glossaries-extra}
\GlsXtrLoadResources[
   src={terms,entries},
]
4.7 Dual Entry Types

type={index},
label-prefix={idx.},
dual-prefix={gls.},
combine-dual-locations={primary},
dual-type={main}
]

\begin{document}
\gls{gls.array}, \gls{gls.vector}, \gls{gls.set}, \gls{gls.matrix}.
\gls{idx.duck}, \gls{idx.aardvark}, \gls{idx.zebra}.
\renewcommand{\glstreenamefmt}[1]{\textsc{#1}}
\printunsrtglossary[type={main},style={index},nogroupskip]
\renewcommand{\glstreenamefmt}[1]{#1}
\renewcommand{\glstreegroupheaderfmt}[1]{\textbf{#1}}
\printunsrtglossary[type={index},style={mcolindexgroup}]
\end{document}

This uses combine-dual-locations to combine the locations for the primary and dual entries so that they only appear in the index.

To avoid the inconvenience of remembering which prefix to use, you can set up the prefixes with \texttt{\glsxtraddlabelprefix} and reference entries with \texttt{\gls}, \texttt{\Gls} etc instead of \texttt{\gls}, \texttt{\Gls} etc.

@dualindexabbreviation

The @dualindexabbreviation entry type is similar to @dualindexentry and again, by default, the field mapping is:

- name $\rightarrow$ name

If the prefix fields are defined, then the default mappings additionally include:

- prefix $\rightarrow$ dualprefix
- prefixplural $\rightarrow$ dualprefixplural
- prefixfirst $\rightarrow$ dualprefixfirst
- prefixfirstplural $\rightarrow$ dualprefixfirstplural
- dualprefix $\rightarrow$ prefix
- dualprefixplural $\rightarrow$ prefixplural
4.7 Dual Entry Types

- **dualprefixfirst** $\rightarrow$ **prefixfirst**
- **dualprefixfirstplural** $\rightarrow$ **prefixfirstplural**

However in this case the required fields are **short** and **long**. The **name** for the primary entry defaults to **short** if omitted. (This may be changed with the **abbreviation-name-fallback** option.) The fallback for the **sort** field is given by **abbreviation-sort-fallback**, which defaults to the **short** field (see section 5.8).

For example:

```plaintext
@dualindexabbreviation{html,
  short = {HTML},
  long = {hypertext markup language}
}
```

is like:

```plaintext
@index{html,name={HTML}}
@abbreviation{dual.html,
  short = {HTML},
  long = {hypertext markup language}
}
```

The primary term is defined using \bibglsnewdualindexabbreviation, which encapsulates the **name** to match the font used by the dual abbreviation. The encapsulation command depends on the **abbreviation-name-fallback** value. If it’s the **short** field then \bibglsuseabbrevfont is used, otherwise \bibglsuselongfont is used.

The primary definition also by default sets the **category** to index (although this again may be overridden). The dual term is defined using \bibglsnewdualindexabbreviation-secondary.

@**dualindexsymbol**

The @**dualindexsymbol** entry type is similar to @**dualindexentry**, but by default the field mappings are:

- **symbol** $\rightarrow$ **name**
- **name** $\rightarrow$ **symbol**
- **symbolplural** $\rightarrow$ **plural**
- **plural** $\rightarrow$ **symbolplural**

If the prefix fields are defined, then the default mappings additionally include:

- **prefix** $\rightarrow$ **dualprefix**
4.7 Dual Entry Types

- prefixplural $\mapsto dualprefixplural$
- prefixfirst $\mapsto dualprefixfirst$
- prefixfirstplural $\mapsto dualprefixfirstplural$
- dualprefix $\mapsto prefix$
- dualprefixplural $\mapsto prefixplural$
- dualprefixfirst $\mapsto prefixfirst$
- dualprefixfirstplural $\mapsto prefixfirstplural$

The required field is: symbol. If the name field is omitted, the dual entry is assigned a symbol from the original (unprefixed) label. The primary entries are defined using \texttt{\bibglsnewdualindexsymbol}, which by default sets the category to index, and the dual entries are defined using \texttt{\bibglsnewdualindexsymbolsecondary}, which by default sets the category to symbol. For example:

\begin{verbatim}
@dualindexsymbol{pi, 
    symbol={\ensuremath{\pi}}, 
    description={ratio of a circle's circumference to its diameter} 
}
\end{verbatim}

is like:

\begin{verbatim}
@index{pi,symbol={\ensuremath{\pi}}}

@symbol{dual.pi, 
    name={\ensuremath{\pi}}, 
    symbol={pi}, 
    description={ratio of a circle's circumference to its diameter} 
}
\end{verbatim}

For example, suppose I have a file called symbols.bib that contains:

\begin{verbatim}
@dualindexsymbol{pi, 
    symbol={\ensuremath{\pi}}, 
    description={ratio of a circle's circumference to its diameter} 
}
\end{verbatim}

\begin{verbatim}
@dualindexsymbol{e, 
    name={Euler's number}, 
    symbol={\ensuremath{e}}, 
    description={base of the natural logarithm} 
}
\end{verbatim}
4.7 Dual Entry Types

Then the previous example document can be modified to have an index, a glossary and a list of symbols:

```latex
\documentclass{report}
\usepackage[colorlinks]{hyperref}
\usepackage[record,symbols,index,stylemods={mcols}]{glossaries-extra}
\newcommand\bibglsnewdualindexsymbolsecondary[5]{\longnewglossaryentry*{#1}{name={#3},category=symbol,%
symbol={#4},#2,type={symbols}}{#5}}
\newcommand\indexprimary[1]{\glsadd[format={hyperbf}]{idx.#1}}
\glsdefpostdesc{symbol}{\indexprimary{\glscurrententrylabel}}
\glsdefpostdesc{general}{\indexprimary{\glscurrententrylabel}}
\GlsXtrLoadResources[
src={entries,terms,symbols},
type={index},
set-widest,  
label-prefix={idx.},
dual-prefix={},
combine-dual-locations={primary},
dual-sort={letter-case},
dual-type={main}]
\glsxtrnewglslike[hyper={false}]{idx.}{\idx}{\idxpl}{\Idx}{\Idxpl}
\begin{document}
\gls{array}, \gls{vector}, \gls{set}, \gls{matrix}.
\idx{duck}, \idx{aardvark}, \idx{zebra}.
\gls{e} and \gls{pi}.
\newpage
\gls{array}, \idx{vector}, \idx{set}, \gls{matrix}.
\newpage
\gls{array}, \gls{vector}, \gls{set}, \gls{matrix}.
\renewcommand{\glstreenamefmt}[1]{\textsc{#1}}
\printunsrtglossary[type={main},nogroupskip,style={alttree}]
```

85
Here I’ve provided some convenient commands for referencing the primary (index) terms \idx, \idxpl, \Idx and \Idxpl. This means I don’t need to worry about the label prefix and it also switches off the hyperlinks (with \hyper={false}). These custom commands are defined using:

\begin{verbatim}
\glsxtrnewglslike[⟨options⟩]{⟨prefix⟩}{⟨gls-like cs⟩}{⟨Gls-like cs⟩}{⟨Glspl-like cs⟩}
\end{verbatim}

which, in this case, essentially does:

\begin{verbatim}
\newcommand{\idx}[2][]{\gls[hyper={false},#1]{idx.#2}}
\newcommand{\Idx}[2][]{\Gls[hyper={false},#1]{idx.#2}}
\newcommand{\idxpl}[2][]{\glspl[hyper={false},#1]{idx.#2}}
\newcommand{\Idxpl}[2][]{\Glspl[hyper={false},#1]{idx.#2}}
\end{verbatim}

but the new commands will also recognise the \gls modifiers, so \idx+ will behave like \gls+ which wouldn’t be possible if \idx was defined using \newcommand in the above manner. There’s a similar command:

\begin{verbatim}
\glsxtrnewgls[⟨options⟩]{⟨prefix⟩}{⟨cs⟩}
\end{verbatim}

if no case-changing versions are required.

I’ve also redefined \bibglsnewdualindexsymbolsecondary to put the dual entries created with @dualindexsymbol into the symbols glossary (which is created with the symbols package option), so it overrides the dual-type={main} setting.

This command also sets the category to symbol, so I can redefine the post-description hook for symbols (\glsxtrpostdescsymbol) to automatically index the symbol definition. Similarly for the general post-description hook \glsxtrpostdescgeneral.

Since the post-description hook isn’t done until the glossary has been created, this requires a slightly longer build process. If the document file is called myDoc.tex, then the complete document build is:

\begin{verbatim}
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
\end{verbatim}
As from glossaries-extra-bib2gls version 1.37, an alternative method is to identify possible label prefixes with \glsxtraddlabelprefix or \glsxtrprependlabelprefix and use \dgls, \dglspl, \dGls or \dGlspl. See the glossaries-extra user manual [13] for further details.

@dualindexnumber

The @dualindexnumber entry type is almost identical to @dualindexsymbol, but the primary entries are defined using \bibglsnewdualindexnumber, which by default sets the category to index, and the dual entries are defined using \bibglsnewdualindexnumber-secondary, which by default sets the category to number.

@dualabbreviationentry

The @dualabbreviationentry entry type is similar to @dualentry, but by default the field mappings are:

- \texttt{long} → \texttt{name}
- \texttt{longplural} → \texttt{plural}
- \texttt{short} → \texttt{text}

If the prefix fields are defined, then the default mappings additionally include:

- \texttt{prefix} → \texttt{dualprefix}
- \texttt{prefixplural} → \texttt{dualprefixplural}
- \texttt{prefixfirst} → \texttt{dualprefixfirst}
- \texttt{prefixfirstplural} → \texttt{dualprefixfirstplural}
- \texttt{dualprefix} → \texttt{prefix}
- \texttt{dualprefixplural} → \texttt{prefixplural}
- \texttt{dualprefixfirst} → \texttt{prefixfirst}
- \texttt{dualprefixfirstplural} → \texttt{prefixfirstplural}

You may need to add a mapping from \texttt{shortplural} to \texttt{plural} if the default is inappropriate. (In bib2gls version 1.0 this entry type was originally called @dualentryabbreviation. In version 1.1, it was renamed @dualabbreviationentry which makes for a more consistent naming scheme @\texttt{dual}\{primary\}\{dual\}.)

The required fields are: \texttt{short}, \texttt{long} and description. This entry type is designed to emulate the example \texttt{\newdualentry} command given in the glossaries user manual [14]. The primary entry is an abbreviation with the given \texttt{short} and \texttt{long} fields (but not the
4.7 Dual Entry Types

description) and the secondary entry is a regular entry with the name copied from the long field. The fallback for the sort is given by abbreviation-sort-fallback, which defaults to the short field (see section 5.8).

For example:

@dualabbreviationentry{svm,  
  long = {support vector machine},  
  short = {SVM},  
  description = {statistical pattern recognition technique}  
}

is rather like doing:

@abbreviation{svm,  
  long = {support vector machine},  
  short = {SVM}  
}

@entry{dual.svm,  
  name = {support vector machine},  
  description = {statistical pattern recognition technique}  
}

but dual.svm will automatically be selected if svm is indexed in the document. If dual.svm isn’t explicitly indexed, it won’t have a location list.

If the sort field is missing bib2gls by default falls back on the name field. If this is missing, this sort value will fallback on the short field. This means that if name isn’t explicitly given in @dualabbreviationentry, then the primary entry will be sorted according to short but the dual will be sorted according its name (which has been copied from the primary long).

Entries provided using @dualabbreviationentry will be defined with:

\bibglsnewdualabbreviationentry

(which uses \newabbreviation) for the primary entries and with:

\bibglsnewdualabbreviationentrysecondary

(which uses \longnewglossaryentry) for the secondary entries. This means that if the abbreviations package option is used, the primary entry will be put in the abbreviations glossary and the secondary entry in the main glossary. Use the type and dual-type options to override this.

@dualentryabbreviation

This entry type is deprecated as from bib2gls version 1.1. It’s functionally equivalent to @dualabbreviationentry but its name doesn’t fit the general dual entry naming scheme.
4.7 Dual Entry Types

@dualsymbol

This is like @dualentry but the default mappings are:

- name $\mapsto$ symbol
- plural $\mapsto$ symbolplural
- symbol $\mapsto$ name
- symbolplural $\mapsto$ plural

If the prefix fields are defined, then the default mappings additionally include:

- prefix $\mapsto$ dualprefix
- prefixplural $\mapsto$ dualprefixplural
- prefixfirst $\mapsto$ dualprefixfirst
- prefixfirstplural $\mapsto$ dualprefixfirstplural
- dualprefix $\mapsto$ prefix
- dualprefixplural $\mapsto$ prefixplural
- dualprefixfirst $\mapsto$ prefixfirst
- dualprefixfirstplural $\mapsto$ prefixfirstplural

The name and symbol fields are required. For example:

@dualsymbol{pi,
    name={pi},
    symbol={\ensuremath{\pi}},
    description={the ratio of the length of the circumference of a circle to its diameter}}

Entries are defined using \bibglsnewdualsymbol, which by default sets the category to symbol.

@dualnumber

This is almost identical to @dualsymbol but entries are defined using \bibglsnewdualnumber, which by default sets the category to number.

The above example could be defined as a number since $\pi$ is a constant:
4.7 Dual Entry Types

@dualnumber{pi,
    name={pi},
    symbol={\ensuremath{\pi}},
    description={the ratio of the length of the circumference of a circle to its diameter},
    user1={3.14159}
}

This has stored the approximate value in the user1 field. The post-description hook could then be adapted to show this.

\glsdefpostdesc{number}{%
    \ifglshasfield{user1}{\glscurrententrylabel}
    { (approximate value: \glscurrentfieldvalue)}%
    {}%
}

This use of the user1 field means that the dual entries could be sorted numerically according to the approximate value:

\usepackage[record,postdot,numbers,style={index}]{glossaries-extra}

\GlsXtrLoadResources[
    src={entries},% entries.bib
dual-type={numbers},
dual-sort={double},% decimal sort
dual-sort-field={user1}
]

@dualabbreviation

The @dualabbreviation entry type is similar to @dualentry, but by default the field mappings are:

- short → dualshort
- shortplural → dualshortplural
- long → duallong
- longplural → duallongplural
- dualshort → short
- dualshortplural → shortplural
- duallong → long
4.7 Dual Entry Types

- duallongplural ➔ longplural

If the prefix fields are defined, then the default mappings additionally include:

- prefix ➔ dualprefix
- prefixplural ➔ dualprefixplural
- prefixfirst ➔ dualprefixfirst
- prefixfirstplural ➔ dualprefixfirstplural
- dualprefix ➔ prefix
- dualprefixplural ➔ prefixplural
- dualprefixfirst ➔ prefixfirst
- dualprefixfirstplural ➔ prefixfirstplural

The required fields are: short, long, dualshort and duallong. This includes some new fields: dualshort, dualshortplural, duallong and duallongplural. If these aren’t already defined, they will be provided in the .glstex file with

\glsxtrprovidestoragekey{⟨key⟩}{key}{key}

Note that this use with an empty third argument prevents the creation of a field access command (analogous to \glsentrytext). The value can be accessed with \glsxtrusefield instead. Remember that the field won’t be available until the .glstex file has been created.

Note that bib2gls doesn’t know what abbreviation styles are in used, so if the sort field is missing it will fallback on the short field. If the abbreviations need to be sorted according to the long field instead, use abbreviation-sort-fallback=long (see section 5.8).

Terms that are defined using @dualabbreviation will be written to the output file using \bibglsnewdualabbreviation.

If the dual-abbrv-backlink option is on, the default field used for the backlinks is the dualshort field, so you’ll need to make sure you adapt the glossary style to show that field.

The simplest way to do this is through the category post-description hook.

For example, if the entries all have the category set to abbreviation, then this requires redefining \glsxtrpostdescabbreviation (either with \renewcommand or via \glsdef-postdesc).

Here’s an example dual abbreviation for a document where English is the primary language and German is the secondary language:

@dualabbreviation{rna,
  short={RNA},
  dualshort={RNS},
  long={ribonucleic acid},
  duallong={Ribonukleinsäure}
}
4.7 Dual Entry Types

If the abbreviation is in the file called entries-dual-abbrv.bib, then here's an example document:

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[ngerman,main=english]{babel}
\usepackage[colorlinks]{hyperref}
\usepackage[record,nomain]{glossaries-extra}
\newglossary*{english}{English}
\newglossary*{german}{German}
\setabbreviationstyle{long-short}
\glsdefpostdesc{abbreviation}{%
  \ifglshasfield{dualshort}{\glscurrententrylabel}{}% 
  \space(\glscurrentfieldvalue)%
}\%
%
\GlsXtrLoadResources[
src={entries-dual-abbrv},% entries-dual-abbrv.bib
type={english},% put primary entries in glossary 'english'
dual-type={german},% put dual entries in glossary 'german'
label-prefix={en.},% primary label prefix
dual-prefix={de.},% dual label prefix
sort={en},% sort primary entries according to language 'en'
dual-sort={de-1996},% sort dual entries according to 'de-1996'
% (German new orthography)
dual-abbrv-backlink% add links in the glossary to the opposite entry
]
\begin{document}

English: \gls{en.rna}; \gls{en.rna}. 

German: \gls{de.rna}; \gls{de.rna}. 

\printunsrtglossaries

92
If the label-prefix is omitted, then only the dual entries will have a prefix:

English: \gls{rna}; \gls{rna}.

German: \gls{de.rna}; \gls{de.rna}.

Another variation is to use the long-short-user abbreviation style and modify the associated \glsxtruserfield so that the duallong field is selected for the parenthetical material:

\renewcommand*{\glsxtruserfield}{duallong}

This means that the first use of the primary entry is displayed as

- ribonucleic acid (RNA, Ribonukleinsäure)

and the first use of the dual entry is displayed as:

- Ribonukleinsäure (RNS, ribonucleic acid)

Here’s an example to be used with the long-short-desc style:

@dualabbreviation{rna,
    short={RNA},
    dualshort={RNS},
    long={ribonucleic acid},
    duallong={Ribonukleinsäure}
    description={a polymeric molecule},
    user1={Ein polymeres Molekül}
}

This stores the dual description in the user1 field, so this needs a mapping. The new example document is much the same as the previous one, except that the dual-abbrv-map option is needed to include the mapping between the description and user1 fields:

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[ngerman,main=english]{babel}
\usepackage[hyperref]{colorlinks}
\usepackage[record,nomain]{glossaries-extra}
\newglossary*{english}{English}
\newglossary*{german}{German}
4.7 Dual Entry Types

\setabbreviationstyle{long-short-desc}

\glsdefpostdesc{abbreviation}{%
    \ifglshasfield{dualshort}{\glscurrententrylabel}
    {%
        \space(\glscurrentfieldvalue)%
    }%
}%

\GlsXtrLoadResources[
    src={entries-dual-abbrv-desc},% entries-dual-abbrv-desc.bib
    type={english},% put primary entries in glossary 'english'
    dual-type={german},% put dual entries in glossary 'german'
    label-prefix={en.},% primary label prefix
    dual-prefix={de.},% dual label prefix
    sort={en},% sort primary entries according to language 'en'
    abbreviation-sort-fallback={long},% fallback on 'long' field
    dual-sort={de-1996},% sort dual entries according to 'de-1996'
    % (German new orthography)
    dual-abbrv-backlink,% add links in the glossary to the opposite entry
    % dual key mappings:
    dual-abbrv-map={%
        {short,shortplural,long,longplural,dualshort,dualshortplural,
            duallong,duallongplural,description,user1},
        {dualshort,dualshortplural,duallong,duallongplural,short,shortplural,
            long,longplural,user1,description}
    }%
]

\begin{document}

English: \gls{en.rna}; \gls{en.rna}.

German: \gls{de.rna}; \gls{de.rna}.

\printunsrtglossaries
\end{document}

Note that since this document uses the long-short-desc abbreviation style, the abbreviation -sort-fallback needs to be changed to long.
If I change the order of the mapping to:

\beginverbatim
dual-abbrv-map={%
    {long,longplural,short,shortplural,dualshort,dualshortplural,

4.8 Tertiary Entry Types

A tertiary entry type is essentially a dual entry that creates three separate (but related) glossaries-extra entry definitions per .bib entry. As with dual entries, the first of these is the primary entry. The second and third are referred to as the secondary entry and tertiary entry.

The tertiary entry is effectively an appendage of the secondary entry-secondary, and is defined by the same associated \bibglsnew...secondary command that defines the secondary entry. Therefore the secondary and tertiary are both considered the dual and are treated as a single entry for the purposes of sorting and collating.
The tertiary entry will never have any locations. Any records found will be assigned to the secondary (and may then be moved to the primary with combine-dual-locations={primary}). The tertiary will always have the same order as the secondary and will have the same group value. You can set the type for the tertiary with tertiary-type and the category with tertiary-category. The label prefix defaults to tertiary and can be changed with tertiary-prefix.

`@tertiaryindexabbreviationentry`

This entry type is very similar to `@dualindexabbreviation` but creates a tertiary entry as well. The required fields are: short and long (as for `@dualindexabbreviation`) and also description. The mappings are shared by both entry types. For example:

```latex
\newglossaryentry{html,
 short = {HTML},
 long = {hypertext markup language},
 description = {a markup language for creating web pages}
}
```

is analogous to:

```latex
\newglossaryentry{html,name={HTML},description={}}
```

```latex
\newabbreviation{dual.html}{HTML}{hypertext markup language}
```

```latex
\newglossaryentry{tertiary.html,
 name={hypertext markup language},
 description={a markup language for creating web pages}
}
```

The last two are actually defined using one command:

```latex\bibglsnewtertiaryindexabbreviationentrysecondary
{dual.html}% secondary label
{tertiary.html}% tertiary label
{...}% secondary fields
{...}% tertiary fields
{HTML}% primary name
{HTML}% short
{hypertext markup language}% long
{a markup language for creating web pages}% description
\```

The `\bibglsnewtertiaryindexabbreviationentrysecondary` command is provided in the `.glstex` file as:
4.9 Multi-Entry Types

A multi-entry type is an entry that may spawn multiple primary entries. This means that both the main entry and the spawned entries are sorted together along with all the other primary entries. In the case of \spawndualindexentry, the main and spawned entries are primary. The main entry’s dual is created as per \dualindexentry.

@bibtexentry

The @bibtexentry type will typically need to be aliased as it’s designed for converting \BibTeX entries into \bib2gls entries. For example, to make \bib2gls treat @article and @book as though they were both @bibtexentry:

```latex
entry-type-aliases=
  article=bibtexentry,
  book=bibtexentry
```

For convenience, glossaries-extra-bib2gls v1.29+ provides \GlsXtrBibTeXEntryAliases which covers all the standard \BibTeX entry types. Alternatively, you can use unknown-entry -alias={bibtexentry} to alias all entries that aren’t recognised by bib2gls. If you use category={same as original entry}, the category field will be set to the original entry type (for example, article or book). Similarly you can use type={same as original entry} to set the type field (but remember that the glossary types will need to be defined in the document).

There are no required fields. The fallback for the sort field is given by \bibtexentry-sort -fallback (see section 5.8). If you want to access any of the \BibTeX fields, you will need to alias or define them. For example:

```latex
field-aliases=
  title=name
```
4.9 Multi-Entry Types

Since B\TeX{}’s \texttt{type} field conflicts with \texttt{bib2gls}'s \texttt{type} field, when \texttt{bib2gls} parses \texttt{@bibtexentry} it will convert \texttt{type} to \texttt{bibtetype}, so you must use \texttt{bibtetype} as the identifier when aliasing.

Alternatively, you can use \texttt{\GlsXtrProvideBibTeXFields} which uses \texttt{\glsaddstorage-key} to provide all the standard B\TeX{} fields. (Remember that new fields must be defined before the first resource set.)

The \texttt{@bibtexentry} essentially creates an \texttt{@index} form of entry, but it additionally defines a \texttt{@contributor} entry for each listed author or editor and updates the dependency lists: each \texttt{@contributor} is added to the main \texttt{@bibtexentry}’s list of dependencies (so if the \texttt{@bibtexentry} has a record then all its satellite \texttt{@contributors} are selected with the default \texttt{selection=\{recorded and deps\}}, and each \texttt{@contributor} is treated as having a cross-reference to the main \texttt{@bibtexentry} (so if a \texttt{@contributor} has a record then all the linked \texttt{@bibtexentry} terms will be selected if \texttt{selection=\{recorded and deps and see\}}). You can instruct \texttt{bib2gls} to treat \texttt{\citation} as an ignored record using \texttt{--cite-as-record}.

Each contributor is effectively defined as:

\begin{verbatim}
@contributor{⟨label⟩,
    name={\bibglscontributor{⟨forenames⟩}{⟨von⟩}{⟨surname⟩}{⟨suffix⟩}}}
\end{verbatim}

The label is obtained by converting the \texttt{name} to a label, using the same function as \texttt{labelify} (which means it’s governed by \texttt{labelify-replace}).

The \texttt{author} and \texttt{editor} fields are always checked, even if those fields aren’t recognised by \texttt{bib2gls}, (which they aren’t by default). These checks are performed before field aliases are applied. If neither field is present, no additional entries are spawned. If the dependent \texttt{@contributor} entry has already been defined, it won’t be redefined, but will have the new \texttt{@bibtexentry} added to its internal \texttt{bibtexentry} field.

The main \texttt{@bibtexentry} is defined using \texttt{\bibglsnewbibtexentry} and is followed by:

\begin{verbatim}
\glsxtrfieldlistadd{⟨id⟩}{bibtexcontributor}{⟨contributor-id⟩}
\end{verbatim}

where \texttt{⟨id⟩} is the label identifying the main \texttt{@bibtexentry} and \texttt{⟨contributor-id⟩} is the label identifying the contributor, for each contributor that has been selected.

Each contributor is defined using \texttt{\bibglsnewcontributor}. The definition is followed by:

\begin{verbatim}
\glsxtrfieldlistadd{⟨contributor-id⟩}{bibtexentry}{⟨id⟩}
\glsxtrfieldlistadd{⟨contributor-id⟩}{bibtexentry@⟨entry-type⟩}{⟨id⟩}
\end{verbatim}

for each selected \texttt{@bibtexentry} associated with that contributor. The second line provides the internal list field \texttt{bibtexentry@⟨entry-type⟩}, where \texttt{⟨entry-type⟩} is the original entry type (before it was aliased to \texttt{@bibtexentry} and converted to lower case). For example \texttt{article} or \texttt{book}.

You can iterate over these internal list fields using \texttt{\glsxtrfielddolistloop} or \texttt{\glsxtrfieldforlistloop}. For example:

98
4.9 Multi-Entry Types

(while the resource option `field-aliases={title=name}` has been used). Here's an example that uses the test `xampl.bib` file that's provided with TeX distributions:

\documentclass{article}

\usepackage[record,nomain]{glossaries-extra}

\newglossary*{contributors}{Authors/Editors}
\newglossary*{titles}{Titles}

\newcommand{\bibglsnewbibtexentry}[4]{%
  longnewglossaryentry*{\#1}{name=\#3,\#2,type={titles}}{\#4}%
}

\GlsXtrLoadResources[
  src={xampl},
  write-preamble={false},
  entry-type-aliases={
    \GlsXtrBibTeXEntryAliases
  },
  field-aliases={
    title=name
  },
  replicate-fields={
    note=name
  },
  labelify-replace={
    {{[ \string\-\string\.]}}{}
  },
  type={contributors},
  category={same as original entry},
  sort-field={category},
  sort-suffix={name}
]
4.9 Multi-Entry Types

\glsxtrsetgrouptitle{article}{Articles}
\glsxtrsetgrouptitle{booklet}{Booklets}
\glsxtrsetgrouptitle{book}{Books}
\glsxtrsetgrouptitle{inbook}{Book Chapters}
\glsxtrsetgrouptitle{misc}{Miscellaneous}

\newcommand{\contributorhandler}[1]{\par\glsentryname{#1} (#1)}

\newcommand{\glsxtrpostdesccontributor}{\%}
  \glsxtrifhasfield{bibtexentry}{\glscurrententrylabel}{\%
    \glsxtrfieldforlistloop{\glscurrententrylabel}{bibtexentry}{\contributorhandler}{\%
      \par No titles.}{\%
  \begin{document}
Sample \cite{book-minimal,article-full,inbook-full,misc-minimal}.
Another sample \cite{booklet-minimal,misc-full,article-minimal}.

\bibliographystyle{plain}
\bibliography{xampl}

\printunsrtglossary[type={contributors},style={altlist}]
\printunsrtglossary*[type={titles},style={indexgroup}]
  \% \newcommand{\glsxtrgroupfield}{category} \%
  \renewcommand{\glstreenamefmt}[1]{\textbf{#1}} \%
  \renewcommand{\glstreegroupheaderfmt}[1]{\textbf{#1}} \%
\end{document}

If the file is called myDoc.tex then the document build is:

pdflatex myDoc
bib2gls --cite-as-record myDoc
bibtex myDoc
pdflatex myDoc
pdflatex myDoc
4.9 Multi-Entry Types

@progenitor

The @progenitor type of entries are the only place where the adoptparents field is permitted. The value should be a comma-separated list of labels. The adoptparents field must be set and must contain a least one label. If the value contains any of the characters \ (backslash), { (open brace) or } (close brace) then the field will be interpreted (if the default --interpret settings is on).

Since entries are spawned before fields are processed, the adoptparents field is parsed before any field aliases (field-aliases) or replication (replicate-fields) takes place. However, if the adoptparents field isn’t found, bib2gls will check for a simple mapping in both the field-aliases and replicate-fields settings.

This entry type creates a main progenitor term (with all the given fields except adoptparents) and \( n \) spawned progeny terms, where \( n \) is the number of elements in the adoptparents field, that are dependent on the main term.

Each of the spawned progeny entries have the field identified by adopted-parent-field (parent by default) set to the corresponding element in the adoptparents field.

All fields from the original definition are copied except for the adoptparents, alias and parent fields. The parent field is never copied, regardless of the value of adopted-parent-field. If the adopted parent field is changed to one that’s contained in the original entry, it’s value will be from adoptparents not the value from the original entry.

The copied fields follow the same conditions as normal entries. (For example, unknown fields are ignored, case-changes are applied, if appropriate, and the type field must reference a valid glossary, if set.) If progenitor-type is set, then this assignment is made after the progeny are created and only applies to the main progenitor entry. The type for the progeny can be set with progeny-type. For example, progeny-type={same as parent} will ensure that the progeny are in the same glossary type as their parent entry.

For example, an entry defined as:

```bib2gls
@progenitor{@id, 
  adoptparents = {<parent-1 id>,...,<parent-N id>}, 
  <field-name-1> = {<text>}, 
  ...
  <field-name-n> = {<text>}
}
```

is essentially like:

```bib2gls
@index{@id, 
  progeny = {<parent-1 id>.<id>,...,<parent-N id>.<id>}, 
  <field-name-1> = {<text>}, 
  ...
  <field-name-n> = {<text>}
}
@index{<parent-1 id>.<id>, 
```
progenitor = {{id}},
parent = {{parent-1 id}},
⟨field-name-1⟩ = {{text}},
...
⟨field-name-n⟩ = {{text}}
}

@index{{parent-N id}.⟨id⟩,
progenitor = {{id}},
parent = {{parent-N id}},
⟨field-name-1⟩ = {{text}},
...
⟨field-name-n⟩ = {{text}}
}

This creates the main (progenitor) ⟨id⟩ entry, which contains all the fields (except for adopted-parents) that were in the original @progenitor definition and has the new field progeny set to the comma-separated list of spawned entry labels. The main entries are defined in the .glstex file with \bibglsnewprogenitor.

In addition to the main ⟨id⟩ entry, the above also creates the spawned progeny entries ⟨parent-1 id⟩.⟨id⟩, ..., ⟨parent-N id⟩.⟨id⟩ that are dependent on the main ⟨id⟩ entry.

The spawned entries have the parent field set to the corresponding label obtained from the adoptedparents list. This parent entry must also be defined, as usual for the parent field. (This restriction obviously doesn’t apply if adopted-parent-field is changed from the default parent.) The spawned entries are defined in the .glstex file with \bibglsnewspawnedindex.

If the main progenitor entry is referenced in the document then (assuming the default selection criteria) the spawned entries will also be automatically selected. You can check for the existence of the progenitor field using \glsxtrifhasfield and fetch the location field from the main entry, if required.

Although the spawned entries are considered dependents of the main entry, the reverse doesn’t apply. If a spawned entry is referenced in the document (with ⟨parent-id⟩.⟨id⟩) then the main entry and its other spawned entries aren’t automatically selected.

For example, suppose the file entries.bib contains:

@indexplural{stylesheet, text={stylesheet language}}

@index{webdesign, name={web design}}

@indexplural{markup, text={markup language}}

@progenitor{xml,
  name={XML},
}
4.9 Multi-Entry Types

```latex
\documentclass{article}
\usepackage[record,stylemods={tree},style={index}]{glossaries-extra}
\GlsXtrLoadResources[src={entries},selection={all}]
\newcommand*{\glstreenamefmt}[1]{#1}
\begin{document}
\printunsrtglossaries
\end{document}
```

Then the resulting list will be:

CSS
HTML
markup language
  HTML
  XML
stylesheet language
  CSS
  XSL
web design
  CSS
  HTML
XML
XSL

103
This allows the HTML and CSS entries to be listed under multiple parents.

The following \texttt{@spawn\{single-type\}} commands are all forms of \texttt{@progenitor} that create the given \texttt{@\{single-type\}} of entry. The spawned entries are actually created with the private entry type \texttt{@spawned\{type\}}. In the case of \texttt{@progenitor}, the spawned entries are defined as a \texttt{@spawnedindex\{type\}} entry. These special \texttt{@spawned\{type\}} entry types aren’t intended for use in the .bib file, but if you reference the entry type (for example, with \texttt{category={same as entry}}) you will get \texttt{@spawned\{type\}} as the entry type. The original entry type for the spawned entries is the same as the original entry for the main \texttt{@progenitor} entry.

There is currently only one form of dual \texttt{@progenitor} entry and that’s \texttt{@spawndualindex-entry}. Only the main progenitor entry is a dual entry. The spawned progeny are all \texttt{@index} primary entries.

\texttt{@spawnindex}

As \texttt{@progenitor}, but the main entries are defined in the .glstex file with \texttt{\bibglsnewspawnindex} and the spawned entries are defined with \texttt{\bibglsnewspawnedindex}.

\texttt{@spawnindexplural}

As \texttt{@progenitor}, except that it creates \texttt{@indexplural} terms instead of \texttt{@index}. As with \texttt{@indexplural}, if the \texttt{name} field isn’t set, it’s assigned to the same value as the \texttt{plural} field (or the fallback for the \texttt{plural}, if not defined).

The main entries are defined in the .glstex file with \texttt{\bibglsnewspawnindexplural} and the spawned entries are defined with \texttt{\bibglsnewspawnedindexplural}.

\texttt{@spawnentry}

As \texttt{@progenitor}, except that it creates \texttt{@entry} terms instead of \texttt{@index}. As with \texttt{@entry}, the \texttt{description} field is required and either \texttt{name} or \texttt{parent}.

The main entries are defined in the .glstex file with \texttt{\bibglsnewspawnentry} and the spawned entries are defined with \texttt{\bibglsnewspawnedentry}.

\texttt{@spawnabbreviation}

As \texttt{@progenitor}, except that it creates \texttt{@abbreviation} terms instead of \texttt{@index}. As with \texttt{@abbreviation}, the \texttt{short} and \texttt{long} fields are required.

The main entries are defined in the .glstex file with \texttt{\bibglsnewspawnabbreviation} and the spawned entries are defined with \texttt{\bibglsnewspawnedabbreviation}.

\texttt{@spawnacronym}

As \texttt{@progenitor}, except that it creates \texttt{@acronym} terms instead of \texttt{@index}. As with \texttt{@acronym}, the \texttt{short} and \texttt{long} fields are required.
The main entries are defined in the .glistex file with \bibglsnewspawnacronym and the spawned entries are defined with \bibglsnewspawnedacronym.

@spawnssymbol
As @progenitor, except that it creates @symbol terms instead of @index. As with @symbol, the required fields are name or parent, and the description field is required if the name field is missing.

The main entries are defined in the .glistex file with \bibglsnewspawnedsymbol and the spawned entries are defined with \bibglsnewspawnedsymbol.

@spawnnumber
As @progenitor, except that it creates @number terms instead of @index. As with @number, the required fields are name or parent, and the description field is required if the name field is missing.

The main entries are defined in the .glistex file with \bibglsnewspawnednumber and the spawned entries are defined with \bibglsnewspawnednumber.

@spawndualindexentry
As @progenitor, except that the main (progenitor) entry behaves like @dualindexentry. The spawned progeny behave like @index are so are all considered primary entries. The adoptparents field should therefore reference primary entries with the default adopted parent-field={parent}.

The main primary and secondary (dual) entries are defined in the .glistex file with \bibglsnewspawndualindexentry and \bibglsnewspawndualindexentrysecondary. The spawned progeny are defined with \bibglsnewspawnedindex.

4.10 Compound Entry Sets

A compound entry isn’t an entry in the same sense as the above but corresponds to a multi-entry (compound or combined) set provided by glossaries-extra v1.48+, which is defined by the command \multiglossaryentry (or \providemultiglossaryentry). These are referred to as multi-entries in glossaries-extra but are referred to as compound entries here to avoid confusion with the multi-entry types.

Essentially, a label is defined that refers to a set of labels corresponding to entries that have already been defined. One element in the set is considered the main label. Entry labels may appear in multiple sets.

A compound entry provides a convenient way to apply commands like \gls to multiple entries in one command (such as \mgls). Compound entry labels may only be used in the \mgls-like commands or in a cross-reference field.

For example, consider the following document:
4.10 Compound Entry Sets

\documentclass{article}
\usepackage{hyperref}
\usepackage[record,style={tree}]{glossaries-extra}
\setabbreviationstyle{long-only-short-only}
\renewcommand*{\glsxtronlyname}{% 
  \protect\glslongonlyfont{\the\glslongtok}%}
\newabbreviation{clostridium}{C.}{ Clostridium}
\newglossaryentry{botulinum}{name=botulinum, 
  description={},parent=clostridium}
\newglossaryentry{perfringens}{name=perfringens, 
  description={},parent=clostridium}
\begin{document}
\gls{clostridium} \gls{botulinum}, \gls{clostridium} \gls{perfringens}, \gls{clostridium} \gls{botulinum}.
\end{document}

This produces:

Clostridium botulinum, C. perfringens, C. botulinum.

followed by the glossary. This is very cumbersome. Defining a compound entry label simply provides a shortcut:

\multiglossaryentry{cbot}{clostridium,botulinum}
\multiglossaryentry{cperf}{clostridium,perfringens}

(This has to be done after the entries have been defined.) Now the entries can be more compactly referenced:

\mglsts{cbot}, \mglsts{cperf}, \mglsts{cbot}.

Each compound entry set must contain at least two elements. The main label is the label of the element that is considered the main entry of the set. If the main label isn’t identified in \multiglossaryentry then it’s assumed to be the last element in the set.

In the above example, botulinum is the main label of the cbot set, and perfringens is the main label of the cperf set. In both sets, clostridium is the “other label”. If there are more than two elements in the set then “others” refers to all the elements except for the main label. An entry can be a main label of one set and an other label of another set.

The options, which can be applied to all sets with \multiglossaryentrysetup or to a specific set using the first optional argument of \multiglossaryentry, determine if each element of the list has a separate hyperlink to their own target, or if only the main element
4.10 Compound Entry Sets

should have a hyperlink, or if the entire content of \mgl s should be a single hyperlink to the main entry’s target.

With bib2gls, the entries that form the set should be in .bib files as usual. The compound entry set may either be defined in the document .tex file using \multiglossaryentry (or \providemultiglossaryentry) or they can be defined in the .bib file using @compoundset. Remember that the set can only be defined after the entries that make up the elements of the set have been defined. If any .bib files in a resource set contain @compoundset, the definitions will be added at the end of the .glstex file (using \bibgls-defcompoundset).

If you have multiple resource sets that reuse the same .bib file containing @compoundset then either redefine \bibglsdefcompoundset to use \providemultiglossaryentry or prevent duplicate definitions with compound-write-def={false}.

The elements of the set will still need to be indexed as usual to ensure that they have records to enable selection.

The above example can be converted to bib2gls as follows (compound entries defined in the document .tex file):

\documentclass{article}
\usepackage{hyperref}
\usepackage[record,style={tree}]{glossaries-extra}
\setabbreviationstyle{long-only-short-only}
\renewcommand*{\glsxtronlyname}{%\protect\glslongonlyfont{\the\glslongtok}\%}
\GlsXtrLoadResources[src={bacteria}]
\multiglossaryentry{cbot}{clostridium,botulinum}
\multiglossaryentry{cperf}{clostridium,perfringens}
\begin{document}
\mgl s{cbot}, \mgl s{cperf}, \mgl s{cbot}.
\printunsrtglossary
\end{document}

Note that \multiglossaryentry must come after \GlsXtrLoadResources.

The bacteria.bib contains the definitions in the usual way:

@abbreviation{clostridium, 
short={C.}, 
long={Clostridium}}

@index{botulinum, 
parent={clostridium}}

@index{perfringens, 
parent={clostridium}}

107
4.10 Compound Entry Sets

Alternatively, the compound entries can be defined in the .bib file instead:

```latex
@compoundset{cbot,
  elements={clostridium,botulinum}
}
@compoundset{cperf,
  elements={clostridium,perfringens}
}
```

The `\multiglossaryentry` commands should now be removed from the .tex file.

There’s a difference between these two methods on the first LaTeX build. In the first example, `cbot` is known, so `\gls{cbot}` can perform `\gls{clostridium} \gls{botulinum}`. These commands aren’t yet defined so they are both replaced by “???” (resulting in “??? ???”). As usual, the location list is unreliable until entries are defined and the unknown markers “??” can be replaced with the correct content. If the document is in a file called `myDoc.tex` then the document build:

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

will have locations in the resulting PDF file, but they may be incorrect if the associated temporary files were initially missing.

In the second example, `cbot` is unknown, so `\gls{cbot}` is simply displayed as “???”. In this case, the .aux file contains information that `cbot` has been referenced, but there are no associated records. The entries that belong to the `cbot` set will be selected as they are considered dependent on the compound entry. In this case, if you are starting from scratch (no associated temporary files), you will need:

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

At this point, the location lists will appear. After that, you can reduce the document build to:

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

(Until you later add new entries.)

If you don’t want locations for the other elements then set the `ENCAP` to `glsignore`:

```
\multiglossaryentrysetup{encapothers=glsignore}
```
@compoundset

The following fields are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements</td>
<td>The comma-separated list of element labels. This corresponds to the final argument of \multiglossaryentry. (Required.)</td>
</tr>
<tr>
<td>main</td>
<td>The main label. This field is optional. If omitted, the main label is assumed to be the last element.</td>
</tr>
<tr>
<td>option</td>
<td>A comma-separated list of options. This corresponds to the first optional argument of \multiglossaryentry. This field may be omitted.</td>
</tr>
</tbody>
</table>

These fields can only be used in this entry type. Most resource options don't apply to this entry type. Options specific to compound entries are listed in section 5.16.
5 Resource File Options

Make sure that you use glossaries-extra with the record package option. This ensures that bib2gls can pick up the required information from the .aux file, and both record={only} and record={nameref} additionally load the supplementary glossaries-extra–bib2gls package. These two record option values also switch on the sort={none} package option (if you have a new enough version of the base glossaries package), which means that there’s no attempt to assign or process the sort key if it’s omitted from \newglossaryentry (or similar commands). The sort key will be provided by bib2gls for informational purposes, but there’s no need for \LaTeX{} to write it to any external files (unless you use record={hybrid}, in which case you need to prevent bib2gls from sorting using the sort={none} resource option).

The .glstex resource files created by bib2gls are loaded in the document using

\begin{verbatim}
\glsxtrresourcefile[⟨options⟩]{⟨filename⟩}
\end{verbatim}

where ⟨filename⟩ is the name of the resource file without the .glstex extension. You can have multiple \glsxtrresourcefile commands within your document, but each ⟨filename⟩ must be unique, otherwise \LaTeX{} would attempt to input the same .glstex file multiple times (bib2gls checks for non-unique file names). The associated data for each resource file is called the resource set (see section 1.4).

There’s a shortcut command that uses \jobname{} in the ⟨filename⟩:

\begin{verbatim}
\GlsXtrLoadResources[⟨options⟩]
\end{verbatim}

The first instance of this command is equivalent to:

\begin{verbatim}
\glsxtrresourcefile[⟨options⟩]{\jobname}
\end{verbatim}

Any additional use of \GlsXtrLoadResources is equivalent to:

\begin{verbatim}
\glsxtrresourcefile[⟨options⟩]{\jobname-⟨n⟩}
\end{verbatim}

where ⟨n⟩ is number. For example:

\begin{verbatim}
\GlsXtrLoadResources[src={entries-en},sort={en}]
\GlsXtrLoadResources[src={entries-fr},sort={fr}]
\GlsXtrLoadResources[src={entries-de},sort={de-1996}]
\end{verbatim}

This is equivalent to:

\begin{verbatim}
\glsxtrresourcefile[src={entries-en},sort={en}]{\jobname}
\glsxtrresourcefile[src={entries-fr},sort={fr}]{\jobname-1}
\glsxtrresourcefile[src={entries-de},sort={de-1996}]{\jobname-2}
\end{verbatim}
In general, it’s simplest just to use \GlsXtrLoadResources.

The optional argument \texttt{\langle options\rangle} is a comma-separated key=value list. Allowed options are listed below. The option list applies only to that specific \texttt{\langle filename\rangle.glstex} and are not carried over to the next instance of \texttt{\glsxtrresourcefile}. Only the definitions provided in \texttt{@preamble} (if the interpreter is on and \texttt{interpret-preamble={true}}) are carried over to the next resource set and, possibly, cross-resource references if permitted (see section 1.4). The glossaries-extra package doesn’t parse the options, but just writes the information to the .aux file. This means that any invalid options will be reported by bib2gls not by glossaries-extra.

As from glossaries-extra v1.40 you can provide a default set of options by redefining:

\GlsXtrDefaultResourceOptions

This command will be inserted at the start of the options list for all resource commands (and will expand as it’s written to the .aux file). For example:

\begin{verbatim}
\renewcommand{\GlsXtrDefaultResourceOptions}{% 
  selection={all},src={entries}}
\end{verbatim}

\GlsXtrLoadResources[
  type={symbols},
  match={entrytype=symbol}]
\GlsXtrLoadResources[
  type={abbreviations},
  match={entrytype=abbreviation}]

This acts like:

\GlsXtrLoadResources[
  selection={all},src={entries},
  type={symbols},
  match={entrytype=symbol}]
\GlsXtrLoadResources[
  selection={all},src={entries},
  type={abbreviations},
  match={entrytype=abbreviation}]

If you have multiple .bib files you can either select them all using \texttt{src=\langle bib list\rangle} in a single \texttt{\glsxtrresourcefile} call, if they all require the same settings, or you can load them separately with different settings applied.

For example, if the files \texttt{entries-terms.bib} and \texttt{entries-symbols.bib} have the same settings:

\GlsXtrLoadResources[\texttt{src=entries-terms,entries-symbols}]  

Alternatively, if they have different settings:

\GlsXtrLoadResources[\texttt{src=entries-terms,\texttt{type=main}}]  
\GlsXtrLoadResources[\texttt{src=entries-symbols,\texttt{sort=use,\texttt{type=symbols}}]
5 Resource File Options

Note that the sorting is applied to each resource set independently of other resource sets. This means that if you have multiple instances of \glsxtrresourcefile but only one glossary type, the glossary will effectively contain blocks of sorted entries. For example, if file1.bib contains:

\@index{duck}
\@index{zebra}
\@index{aardvark}

and file2.bib contains:

\@index{caterpillar}
\@index{bee}
\@index{wombat}

then

\GlsXtrLoadResources[src={file1,file2}]

will result in the list: aardvark, bee, caterpillar, duck, wombat, zebra. These six entries are all defined when \jobname.glstex is read. Whereas

\GlsXtrLoadResources[src={file1}]
\GlsXtrLoadResources[src={file2}]

will result in the list: aardvark, duck, zebra, bee, caterpillar, wombat. The first three (aardvark, duck, zebra) are defined when \jobname.glstex is read. The second three (bee, caterpillar, wombat) are defined when \jobname-1.glstex is read. Since \printunsrtglos- sary simply iterates over all defined entries, this is the ordering used.

Abbreviation styles must be set (using \setabbreviationstyle) before the resource command that selects the abbreviations from the appropriate .bib file, since the entries are defined (through \newabbreviation or \newacronym) when \glsxtrresourcefile inputs the .glstex file. (Similarly for any associated abbreviation style commands that must be set before abbreviations are defined, such as \glsxtrlongshortdescname.)

Note bib2gls allows .bib files that don’t provide any entries. This can be used to provide commands in @preamble. For example, suppose I have defs.bib that just contains:

@preamble{"\providecommand{\strong}[1]{\textbf{\color{red}#1}}\providecommand{\parenswap}[2]{#2 (#1)}}

This provides two commands:

\strong{(text)}

(which sets the font weight and colour) and

\parenswap{(text1)}{(text2)}
(which just displays its second argument followed by the first in parentheses).

Suppose I also have entries.bib that contains:

```latex
@index{example,
    name={\textbf{\textcolor{red}{\parenswap{stuff}{example}}}}
}
@index{sample}
@index{test}
@index{foo}
@index{bar}
```

This contains an entry that requires the commands provided in defs.bib, so to ensure those commands are defined, I can do:

```latex
\GlsXtrLoadResources[src={defs,entries}]
```

Unfortunately this results in the sort value for example being set to redexample (stuff) because the interpreter has detected the provided commands and expanded:

```latex
\textbf{\textcolor{red}{\parenswap{stuff}{example}}}
```

to:

```latex
\textbf{\textcolor{red}{example (stuff)}}
```

It discards font changes, so \textbf{} is ignored, but it doesn’t recognise \textcolor{} and so doesn’t know that the first argument is just the colour specifier and therefore doesn’t discard it. This means that “example (stuff)” is placed between “foo” and “sample” instead of between “bar” and “foo”.

I can prevent the interpreter from parsing \@preamble:

```latex
\GlsXtrLoadResources[src={defs,entries},interpret-preamble={false}]
```

Now when the sort value for example is obtained from:

```latex
\textbf{\textcolor{red}{\parenswap{stuff}{example}}}
```

no expansion occurs (since \textbf{\textcolor{red}{\parenswap{stuff}{example}}} are now unrecognised) so the sort value ends up as: stuffexample which places “example (stuff)” between “sample” and “test”, which is again incorrect.

The best thing to do in this situation is to split the provided commands into two .bib files: one that shouldn’t be interpreted and one that should.

For example, defs-nointerpret.bib:

```latex
@preamble{"\providecommand{\textbf}[1]{\textbf{\textcolor{red}{#1}}}"
}
```

and defs-interpret.bib:

```latex
@preamble{"\providecommand{\parenswap}[2]{#2 (#1)}}
```

113
5.1 String Concatenation

Now the first one can be loaded with `interpret-preamble={false}`:

```latex
\GlsXtrLoadResources[src={defs-nointerpret},interpret-preamble={false}]
```

This creates a `.glstex` file that provides \texttt{strong} but doesn’t define any entries. The other file `defs-interpret.bib` can then be loaded with the default `interpret-preamble={true}`:

```latex
\GlsXtrLoadResources[src={defs-interpret,entries}]
```

The provided commands are remembered by the interpreter, so you can also do:

```latex
\GlsXtrLoadResources[src={defs-interpret}]
\GlsXtrLoadResources[src={entries}]
```

The contents of \texttt{@preamble} are only written to the associated `.glstex` file, but the definitions contained within the \texttt{@preamble} are retained by the interpreter for subsequent resource sets.

## 5.1 String Concatenation

Some resource options allow string concatenation in their syntax. This is where fragments or substrings can be joined together to form a value. This is similar to the way concatenation occurs in `.bib` files, but a different operator is used. In `.bib` files, the concatenation operator is `#` (hash) but, since this is a problematic character to use in the optional argument of \GlsXtrLoadResources, the operator for string concatenation in resource options is `+` (plus).

A string concatenation `⟨element-list⟩` in a resource option has the following syntax:

```latex
⟨element-list⟩ ::= ⟨element-value⟩ | ⟨element-value⟩ + ⟨element-list⟩
⟨element-value⟩ ::= ⟨string⟩ | ⟨field-ref⟩ | ⟨element-quark⟩{⟨element-list⟩} | ⟨match-ref⟩
⟨match-ref⟩ ::= \MGP{⟨group-ref⟩}
⟨group-ref⟩ ::= ⟨index⟩ | ⟨name⟩
⟨string⟩ ::= "⟨tokens⟩" | {⟨tokens⟩}
```

The `⟨field-ref⟩` syntax is described below, and is used to reference a field value. The element quarks `⟨element-quark⟩`, described below) take an `⟨element-list⟩` argument. If the `⟨element-list⟩` argument evaluates to null, they will return null.

Remember that the argument of \GlsXtrLoadResources is expanded as it’s written to the `.aux` file. This means that care must be taken to prevent premature expansion of quarks or any commands that need to be present in a string.

As from glossaries-extra v1.51, the glossaries-extra–bib2gls package (which is automatically loaded with the \texttt{record} option) provides the command \GlsXtrResourceInitEscSequences which will locally redefine these quark commands to expand to their detokenized form. So you can do:
5.1 String Concatenation

\renewcommand*{\glsxtrresourceinit}{%
  \GlsXtrResourceInitEscSequences
%
}

This means that you can simply write the quark in the resource option without needing to use \protect or \string. The remainder of this section assumes that \glsxtrresourceinit has been redefined to use \GlsXtrResourceInitEscSequences, as in the above example.

As with the .bib format, strings ⟨text⟩ can be delimited by braces {⟨text⟩} or double-quotes "⟨text⟩". If you need a literal double-quote (") then either use brace delimiters or use \". If you need the actual \LaTeX accent command \" then use brace delimiters. If you need braces that start and end in different strings then use double-quote delimiters. For example:

```latex
assign-fields={
  first = "\cs{emph}{" + name + "}"
}
```

The ⟨element-list⟩ may just contain a single element, such as a field reference or a constant string, but it must still conform to the element syntax. For example, if you want to use \copy-to-glossary to copy all entries to a specific glossary, such as index, then you will need to markup index as a string. For example:

```latex
copy-to-glossary={"index"}
```

or

```latex
copy-to-glossary={\{index\}}
```

Note that the outer braces are stripped by the resource option parser before the content is parsed as an ⟨element-list⟩. If only a single set of braces was used, those braces would be stripped leaving a bare index, which would be parsed as a field reference.

The element quarks are uppercase tokens that start with a leading backslash. They have no meaning to bib2gls’s interpreter nor are they defined in the \LaTeX document outside of the scope of the resource command (unless they happen to coincidentally be defined by another package or are a custom command). Quarks occur outside of strings. Any escape sequences occurring within a string are considered to be \LaTeX commands.

```latex\CS{⟨element-list⟩}
```

Returns a control sequence with the control sequence name obtained from concatenating ⟨element-list⟩. Note that this is different from \cs which expands to the detokenized control sequence name as the resource options are written to the .aux file.

For example, if the \LaTeX file has:

```latex
\GlsXtrLoadResources[
  assign-fields={
    name = "\cs{foo}{" + user1 + "}"
  }
  ]
```

115
5.1 String Concatenation

then this will expand the options to the \texttt{.aux} file as

\begin{verbatim}
assign-fields={
    name = "\foo{" + user1 + "}"}
\end{verbatim}

Compare this with:

\begin{verbatim}
\GlslXtrLoadResources[
    assign-fields={
        name = \CS { user1 }
    }
]
\end{verbatim}

which will set the \texttt{name} value to \texttt{\langle csname \rangle} (no arguments) where \texttt{\langle csname \rangle} is the value obtained from the \texttt{user1} field for that entry. Note that \texttt{\langle csname \rangle} will need to be defined in the document to ensure that the document compiles without error but will also need to be recognised by \texttt{bib2gls} if the field value needs to be interpreted (such as when obtaining the \texttt{sort} value).

\begin{verbatim}
\MGP{\langle group-ref \rangle}
\end{verbatim}

The \texttt{\langle match-ref \rangle} element should only be used with a regular expression from an associated conditional (see section 5.2). For example, the \texttt{\langle condition \rangle} part of an assignment rule in \texttt{assign-fields}.

If a match was found, \texttt{\MGP} can be used to reference a group within the match. The \texttt{\langle group-ref \rangle} argument may be either an integer (the group index) or the group name. For example, suppose a custom field called \texttt{ordinal} may contain content such as 1st or 10th and I want to encapsulate the suffix part without altering the \texttt{.bib} file. This can be done as follows:

\begin{verbatim}
assign-fields={
    ordinal = \MGP{1} + " \cs{ord}\{" + \MGP{2} + "}" }[
        ordinal=/\cs{d}+(st|nd|rd|th)/ ]
\end{verbatim}

Alternatively, using named groups:

\begin{verbatim}
assign-fields={
    ordinal = \MGP{num} + " \cs{ord}\{" + \MGP{suffix} + "}" }[
        ordinal=/\?<num>\cs{d}+.(st|nd|rd|th)/ ]
\end{verbatim}

Note that the group name shouldn’t be delimited with double-quotes.

\begin{quote}
The \texttt{\MGP} quark (which expands to the \texttt{\MGP} identifier for \texttt{assign-fields}) isn’t the same as \texttt{\Glslcapturedgroup} (which expands to \texttt{\string$}, allowing a dollar character to be written to the \texttt{.aux} file within the replacement part of \texttt{labelify-replace}).
\end{quote}
5.1 String Concatenation

\TRIM{\langle element-list \rangle}

Returns its argument with any leading and trailing spaces removed.

\INTERPRET{\langle element-list \rangle}

Interprets the contents of \langle element-list \rangle using bib2gls’s interpreter and returns the result, which may be an empty string if the content only contains unknown commands.

\LABELIFY{\langle element-list \rangle}

Converts the contents of \langle element-list \rangle into a label string, according to the labelify criteria.

\LABELIFYLIST{\langle element-list \rangle}

Converts the contents of \langle element-list \rangle into a label-list string, according to the labelify-list criteria.

\LEN{\langle element-list \rangle}

When used within an element list, \LEN returns the length of its \langle element-list \rangle argument as a string or null if \langle element-list \rangle evaluates to null. Note that this is different from using \LEN in a numerical condition where the result is always an integer (see section 5.2). This means that \LEN{\langle list1 \rangle} + \LEN{\langle list2 \rangle} performs string concatenation not numerical addition. Instead, use \LEN{\langle list1 \rangle + \langle list2 \rangle} for the combined length.

The length is the detokenised length, for example, if the name field has the value \textbf{\textit{x}} then \LEN{name} will evaluate to the string "8". You can use

\LEN{\INTERPRET{\langle element-list \rangle}}

to find the length without \LaTeX commands.

The quarks below identify case-changing functions. The \langle element-list \rangle argument will be converted using the appropriate function and the result will be returned. If \langle element-list \rangle evaluates to null then null will be returned.

The case-changing functions will use the resource locale, but whether or not bib2gls recognises the correct rules for the locale depends on whether or not the locale is correctly supported by the Java locale provider. The language resource file may provide assistance with case-conversion (see section 1.8). Note that the case-change is performed by bib2gls not by inserting \LaTeX case-changing commands into the code.

- \LC{\langle element-list \rangle} converts \langle element-list \rangle to lower case;
- \UC{\langle element-list \rangle} converts \langle element-list \rangle to upper case;
- \FIRSTLC{\langle element-list \rangle} converts the first letter of \langle element-list \rangle to lower case;
- \FIRSTUC{\langle element-list \rangle} converts the first letter of \langle element-list \rangle to upper case (sentence case);
5.1 String Concatenation

- \TITLE\{\langle element-list\rangle\} converts \langle element-list\rangle to title case.

There is an additional token \NOCHANGE\{\langle element-list\rangle\} which simply evaluates \langle element-list\rangle and returns it unchanged.\footnote{The \NOCHANGE support wasn’t intentional, but was simply a by-product of the original implementation of the case-changing commands.} This isn’t like \NoCaseChange but is more like \@firstofone. There is little need for it so it’s not defined by \GlsXtrResourceInitEscSequences. The only plausible use for it is if you have a class or package that contains something like:

\begin{verbatim}
\newcommand{\mycase}{\NOCHANGE} \\
% later as the result of some condition: \newcommand{\mycase}{\FIRSTUC} \\
% later on: \GlsXtrLoadResources[ assign-fields={
     name=[o] \cs{\mycase}{name},
} % other assignments ... ]
\end{verbatim}

In most cases, it should be possible to achieve the same result with a conditional associated with the resource option or by adjusting the content passed to the resource command. For example:

\begin{verbatim}
\newcommand{\nameassign}{} \\
% later as the result of some condition: \renewcommand{\nameassign}{name=[o]\FIRSTUC{name},} \\
% later on: \GlsXtrLoadResources[ assign-fields={
     \nameassign
} % other assignments ... ]
\end{verbatim}

The field reference (\langle field-ref \rangle) syntax is more complicated:

\begin{verbatim}
\langle field-ref \rangle ::= \langle value-ref \rangle | \langle entry-ref \rangle -> \langle field-ref \rangle \\
\langle entry-ref \rangle ::= self | parent | root \\
\langle value-ref \rangle ::= \langle field-name \rangle | \langle label-ref \rangle \\
\langle label-ref \rangle ::= \langle label-type \rangle -> \langle label-delineator \rangle \\
\langle label-type \rangle ::= entrytype | entrylabel | entrybib \\
\langle label-delineator \rangle ::= original | actual
\end{verbatim}

where \langle field-name \rangle is the required field name. Note that field names (which need to be used in a string concatenation) can’t include any of the concatenation or conditional markup special characters: + [ ] = , < > or ".
5.2 Complex Conditionals

The \texttt{entry-ref} part indicates which entry the referenced field belongs to. The keywords are: \texttt{self} (the entry itself), \texttt{parent} (the entry’s parent), and \texttt{root} (the entry’s hierarchical root, not including the entry itself). Note that with options such as \texttt{assign-fields} the entry’s ancestors must be defined before the entry in the .bib file because their fields can only be referenced after they have been processed. A grandparent entry can be referenced with \texttt{parent -> parent ->}. Since “parent” is also a field name, if the keyword parent is followed by \texttt{-} then the keyword refers to the parent entry otherwise it refers to the \texttt{parent} field.

The special keywords identify values that aren’t normally stored in a field. The keyword must be followed by the \texttt{delineator}, which may be \texttt{original} or \texttt{actual}. Available keywords:

\begin{description}
\item[\texttt{entrytype}] the entry type, without the leading @, where \texttt{original} refers to the original entry type used in the .bib file and \texttt{actual} refers to the actual entry type, which may have changed as a result of \texttt{entry-type-aliases};
\item[\texttt{entrylabel}] the entry label, where \texttt{original} refers to the original label used in the .bib file and \texttt{actual} refers to the actual label, which may have been altered by options such as \texttt{label-prefix};
\item[\texttt{entrybib}] the .bib file the entry was defined in, where \texttt{original} refers to the basename (without the .bib extension, regardless of whether or not it was included in \texttt{src}) and \texttt{actual} refers to the file name (including the extension and path).
\end{description}

If a syntax error occurs, the error message will show how \texttt{bib2gls} has scanned the information so far. For example, in the case of \texttt{assign-fields={parent name}} the message will be:

```
Error: Invalid syntax for option 'assign-fields': Expected one of -> + [ after ' self -> parent', found 'n'
```

This indicates that it has read “parent” as meaning the \texttt{parent} field of the current entry since it isn’t followed by “\texttt{-}”.

5.2 Complex Conditionals

Some options may have a conditional in their value. In certain cases, such as \texttt{match}, the condition is provided as a regular expression, but other conditionals (such as in \texttt{assign-fields}) are complex. This section describes that complex conditional syntax.

The tokens \& and | indicate logical “AND” and “OR”, respectively, and ! indicates negation. Parentheses ( and ) may be used to control the order of precedence. For example,

\begin{verbatim}
\texttt{boolean1} \lor (\texttt{boolean2} \& \lnot \texttt{boolean3})
\end{verbatim}

Available boolean functions are in the form:
5.2 Complex Conditionals

\langle value1 \rangle \langle cmp \rangle \langle value2 \rangle

where \langle value1 \rangle is the left-hand value and \langle value2 \rangle is the right-hand value. The middle \langle cmp \rangle operator identifies the comparison function.

The left-hand \langle value1 \rangle may be a field reference \langle field-ref \rangle or the integer quark \LEN{\langle element-list \rangle} or the concatenate quark \CAT{\langle element-list \rangle}, where \langle field-ref \rangle references a field value and \langle element-list \rangle is an element list, using the same syntax described in section 5.1.

The right-hand \langle value2 \rangle may be a field reference \langle field-ref \rangle or \CAT{\langle element-list \rangle} or \NULL or a constant string ("\langle string \rangle" or \{\langle string \rangle\}) or a number or a regular expression. You can’t use \LEN on the right-hand as a numeric value (but it may occur inside the argument \CAT). You can’t use \NULL or a regular expression on the left-hand side.

Where \langle value1 \rangle is \LEN{\langle element-list \rangle}, the length evaluates to an integer and may only be used in the numerical comparisons. If \langle element-list \rangle is null, then the length will be 0. The \LEN quark can’t be used in the right hand \langle value2 \rangle part of a numerical comparison. Note that if \LEN occurs inside the argument of \CAT then it becomes a string not a number.

\CAT{\langle element-list \rangle}

Where \langle value1 \rangle or \langle value2 \rangle is \CAT{\langle element-list \rangle}, the \langle element-list \rangle will be evaluated and treated as a string, which will be null if \langle element-list \rangle evaluates to null.

\NULL

The null quark may only be used as \langle value2 \rangle for the equality and inequality comparisons. It can’t be used in any other context. Note that the numeric \LEN doesn’t return null.

Where a field value is referenced (\langle field-ref \rangle), if the field value is undefined (either the field isn’t set or the referenced ancestor entry hasn’t been defined) then, if the designated action is “fallback” (for example, \assign-missing-field-action={fallback}), the fallback value is obtained (see section 5.8). If the value is still undefined it will be considered a null value for the purposes of the comparison. Note that if the designated action is “empty” (for example, \assign-missing-field-action={empty}) there will be no null values.

\langle value1 \rangle=\NULL

Evaluates to true if \langle value1 \rangle is null.

\langle value1 \rangle<>\NULL

Evaluates to true if \langle value1 \rangle is not null.

For the remaining comparisons, null values will be treated as an empty string. Once the \langle field-ref \rangle or \CAT references have been evaluated, their returned value will be turned into a detokenized string for the purposes of the comparison.

The detokenized values from a field reference may contain any TAB or newline characters or additional spacing that are present in the .bib file (unless they have already been stripped by other resource options or field assignments). However, redundant spacing in any literal strings ("\langle string \rangle" or \{\langle string \rangle\}) are likely to be lost when the resource options are written to the .aux file.
5.2 Complex Conditionals

\( ⟨value1⟩=/⟨regex⟩/ \)
\( ⟨value1⟩=/⟨regex⟩/i \)

Evaluates to true if the value matches the given anchored regular expression \( ⟨regex⟩ \). If “ \( i \) ” follows the terminating / then the match is case-insensitive. No other modifiers are recognised, but you can use embedded flag expressions, such as \( ?s \) for “single-line” mode.

In the following string comparisons, the right-hand \( ⟨string⟩ \) is a constant string that must be delimited with double-quotes or braces. The comparisons are according to the Unicode code points (not locale-sensitive), but if the string is followed by “ \( i \) ”, a case-insensitive comparison is used.

\( ⟨value1⟩=⟨string⟩ \)
\( ⟨value1⟩=⟨string⟩i \)

Evaluates to true if the value is equal to the string. For example:

```
category="abbreviation"
```

\( ⟨value1⟩<>⟨string⟩ \)
\( ⟨value1⟩<>⟨string⟩i \)

Evaluates to true if the value is not equal to the string.

\( ⟨value1⟩<⟨string⟩ \)
\( ⟨value1⟩<⟨string⟩i \)

Evaluates to true if the value is lexicographically less than the string.

\( ⟨value1⟩<=⟨string⟩ \)
\( ⟨value1⟩<=⟨string⟩i \)

Evaluates to true if the value is lexicographically less than or equal to the string.

\( ⟨value1⟩>=⟨string⟩ \)
\( ⟨value1⟩>=⟨string⟩i \)

Evaluates to true if the value is lexicographically greater than the string.

\( ⟨value1⟩>=⟨string⟩ \)
\( ⟨value1⟩>=⟨string⟩i \)

Evaluates to true if the value is lexicographically greater than or equal to the string.

In the following numerical comparisons, the given \( ⟨number⟩ \) should use “ . ” for the decimal point and no number group separators. If the \( ⟨number⟩ \) doesn’t contain a decimal point or if \( ⟨value1⟩ \) is the \( \\text{LEN}⟨\text{element-list}⟩ \) quark then an integer comparison is assumed. If \( ⟨value1⟩ \) is empty or isn’t numeric it will be treated as 0. The number shouldn’t be delimited by quotes or braces.
5.2 Complex Conditionals

⟨value1⟩=⟨number⟩

Evaluates to true if the value is equal to ⟨number⟩. For example:

LEN{user1}=0.9

This will return true if the user1 field length is 0 and false otherwise. This is because LEN enforces an integer comparison which means that 0.9 is converted to 0. Similarly:

CAT{"0.9"}=0

This will return true because the ⟨number⟩ 0 is an integer which enforces an integer comparison so the string "0.9" will be converted to the number 0. Compare this with:

CAT{"0.9"}=0.0

This will return false because the ⟨number⟩ 0.0 is a decimal, so a decimal comparison will be used.

⟨value1⟩<>⟨number⟩

Evaluates to true if the value is not equal to ⟨number⟩.

⟨value1⟩<⟨number⟩

Evaluates to true if the value is less than ⟨number⟩.

⟨value1⟩<=⟨number⟩

Evaluates to true if the value is less than or equal to ⟨number⟩.

⟨value1⟩>⟨number⟩

Evaluates to true if the value is greater than ⟨number⟩.

⟨value1⟩>=⟨number⟩

Evaluates to true if the value is greater than or equal to ⟨number⟩.

Finally, the following are string comparisons made after evaluating and detokenizing both ⟨value1⟩ and ⟨value2⟩. The comparisons are case-sensitive and according to the Unicode code points (not locale-sensitive).

⟨value1⟩=⟨value2⟩

Evaluates to true if ⟨value1⟩ is equal to ⟨value2⟩. For example:

name = parent -> name

⟨value1⟩<>⟨value2⟩
5.2 Complex Conditionals

Evaluates to true if \( \langle \text{value1} \rangle \) is not equal to \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle \neq \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is lexicographically less than \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle < \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is lexicographically less than or equal to \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle \leq \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is lexicographically greater than \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle > \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is lexicographically greater than or equal to \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle \geq \langle \text{value2} \rangle \)

\( \langle \text{value1} \rangle \not\in \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is a substring of \( \langle \text{value2} \rangle \). If \( \langle \text{value1} \rangle \) is empty or null it’s considered not a substring regardless of the value of \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle \not\in \langle \text{value2} \rangle \)

The negation of the \( \not\in \) test. Evaluates to true if \( \langle \text{value1} \rangle \) is not a substring of \( \langle \text{value2} \rangle \). This is equivalent to:

\( ! \langle \text{value1} \rangle \not\in \langle \text{value2} \rangle \)

\( \langle \text{value1} \rangle \prefixof \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is a prefix of \( \langle \text{value2} \rangle \) (that is, \( \langle \text{value2} \rangle \) starts with \( \langle \text{value1} \rangle \)). If \( \langle \text{value1} \rangle \) is empty or null it’s considered not a prefix regardless of \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle \not\prefixof \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is not a prefix of \( \langle \text{value2} \rangle \). This is equivalent to:

\( ! \langle \text{value1} \rangle \prefixof \langle \text{value2} \rangle \)

\( \langle \text{value1} \rangle \suffixof \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is a suffix of \( \langle \text{value2} \rangle \) (that is, \( \langle \text{value2} \rangle \) ends with \( \langle \text{value1} \rangle \)). If \( \langle \text{value1} \rangle \) is empty or null it’s considered not a suffix regardless of \( \langle \text{value2} \rangle \).

\( \langle \text{value1} \rangle \not\suffixof \langle \text{value2} \rangle \)

Evaluates to true if \( \langle \text{value1} \rangle \) is not a suffix of \( \langle \text{value2} \rangle \). This is equivalent to:

\( ! \langle \text{value1} \rangle \suffixof \langle \text{value2} \rangle \)
5.3 General Options

**charset=⟨encoding-name⟩**

If the character encoding hasn’t been supplied in the .bib file with the encoding comment

\% Encoding: ⟨encoding-name⟩

then you can supply the correct encoding using \texttt{charset=⟨encoding-name⟩}. In general, it’s better to include the encoding in the .bib file where it can also be read by a .bib managing systems, such as JabRef.

See \texttt{--tex-encoding} for the encoding used to write the .glstex file.

**locale=⟨lang tag⟩**

Sets the default locale for the current resource set. In general, it’s best to set this at the start of the resource option list, if required. If not set, the default will be the document locale, if supplied, otherwise the Java locale will be used.

**interpret-preamble=⟨boolean⟩**

This is a boolean option that determines whether or not the interpreter should parse the contents of \texttt{@preamble}. The default is \texttt{true}. If \texttt{false}, the preamble contents will still be written to the .glstex file, but any commands provided in the preamble won’t be recognised by the interpreter (see chapter 2).

Related options are: \texttt{set-widest} (which uses the interpreter to determine the widest name for the alttree style or the glossary-longextra styles), \texttt{interpret-label-fields} (which governs whether or not fields that must only contain a label should be interpreted), \texttt{labelify} (which converts a field into a string suitable for use as a label), and \texttt{labelify-list} (which converts a field into a string suitable for use as a comma-separated list of labels).

**write-preamble=⟨boolean⟩**

This is a boolean option that determines whether or not the preamble should be written to the .glstex file. The default is \texttt{true}. Note that the preamble will still be parsed if \texttt{interpret-preamble=⟨true⟩} even if \texttt{write-preamble=⟨false⟩}. This means it’s possible to provide \texttt{bib2gls} command definitions in \texttt{@preamble} that don’t get seen by \LaTeXX.

**set-widest=⟨boolean⟩**

The alttree glossary style needs to know the widest \texttt{name} (for each level, if hierarchical). This can be set using \texttt{\glsetwidest} provided by the glossary-tree package (or similar commands like \texttt{\glsupdatewidest} provided by glossaries-extra-stylemods), but this requires knowing which name is the widest. Alternatively, one of the iterative commands such as \texttt{\glsFindWidestTopLevelName} can be used, which slows the document build as it has to iterate over all defined entries.
5.3 General Options

The glossary-longextra package, provided with glossaries-extra v1.37+, also needs to know the widest name, but in this case only the top-level entrytop-level is needed. If this has already been found through the commands provided with the alttree style then that value will be used as the default, but you can set another value that’s only used for the glossary-longextra styles with \glslongextraSetWidest.

The glossaries-extra-bib2gls package provides \glsxtrSetWidest, which sets the widest name for those styles that need it. As from version 1.8, bib2gls now checks for the existence of this command and will use it with set-widest to allow for the new styles provided by the glossary-longextra package.

The boolean option set-widest={true} will try to calculate the widest names for each hierarchical level to help remove the need to determine the correct value within the document. Since bib2gls doesn’t know the fonts that will be used in the document or if there are any non-standard commands that aren’t provided in the .bib files preamble, this option may not work. For example, if one entry has the name defined as:

\begin{verbatim}
name={some \Huge huge text}
\end{verbatim}

and another entry has the name defined as:

\begin{verbatim}
name={some \small small text}
\end{verbatim}

then bib2gls will determine that the second name is the widest although the first will actually be wider when it’s rendered in the document.

When using this option, the transcript file will include the message:

Calculated width of ‘⟨text⟩’: ⟨number⟩

where ⟨text⟩ is bib2gls’s interpretation of the contents of the name field and ⟨number⟩ is a rough guide to the width of ⟨text⟩ assuming the operating system’s default serif font. The entry that has the largest ⟨number⟩ is the one that will be selected. This will then be implemented as follows:

- If the type is unknown then:
  - if the interpreter resolves all name fields to the empty string (that is the name fields all consist of unknown commands) then
    * if there are child entries \bibglssetwidestfallback is used,
    * otherwise \bibglssetwidesttoplevelfallback is used;
  - otherwise \bibglssetwidest is used.

- If the type is known then:
  - if the interpreter resolves all name fields for that type to the empty string (that is the name fields all consist of unknown commands) then
    * if there are child entries \bibglssetwidestfortypefallback is used,
    * otherwise \bibglssetwidesttoplevelfortypefallback is used;

• If the type is unknown then:
  - if the interpreter resolves all name fields to the empty string (that is the name fields all consist of unknown commands) then
    * if there are child entries \bibglssetwidestfallback is used,
    * otherwise \bibglssetwidesttoplevelfallback is used;

• If the type is known then:
  - if the interpreter resolves all name fields for that type to the empty string (that is the name fields all consist of unknown commands) then
    * if there are child entries \bibglssetwidestfortypefallback is used,
    * otherwise \bibglssetwidesttoplevelfortypefallback is used;
5.3 General Options

– otherwise \bibglstestwidestfor{type} is used.

This leaves \TeX to compute the width according to the document fonts. If \bib2gls can’t correctly determine the widest entry then you will need to use one of the commands provided by \gls{-}tree, \gls{-}longextra or \gls{-}extra-new\-stylemods to set it.

In general, if you have more than one glossary it’s best to set the type using options like type and dual-type if you use set-widest.

entry-type-aliases=⟨key=value list⟩

In the \bib file, the data is identified by @⟨entry-type⟩, such as @abbreviation. It may be that you want to replace all instances of @⟨entry-type⟩ with a different type of entry. For example, suppose my \bib file contains abbreviations defined in the form:

@abbreviation{html, 
    short = {html},
    long = {hypertext markup language},
    description = {a markup language for creating web pages}
}

but suppose in one of my documents I actually want all these abbreviations defined with @dualabbreviationentry instead of @abbreviation. Instead of editing the \bib file I can just supply a mapping:

\GlsXtrLoadResources[ 
  src={entries},% data in entries.bib 
  entry-type-aliases={abbreviation=dualabbreviationentry}
]

This makes all instances of @abbreviation behave as @dualabbreviationentry. You can have more than one mapping. For example:

\GlsXtrLoadResources[ 
  src={entries},% data in entries.bib 
  entry-type-aliases={ 
    % @abbreviation -> @dualabbreviationentry: 
    abbreviation=dualabbreviationentry, 
    % @entry -> @index: 
    entry=index 
  }
]

This option isn’t cumulative. Multiple instances of entry-type-aliases override previous instances. If ⟨key=value list⟩ is empty there will be no mappings. You can save the original entry type in the originalentrytype field with save-original-entrytype.

Here’s another example entry in a \bib file:
Ordinarily this entry would be ignored since @foo isn’t recognised, but it can be mapped like this:

\GlsXtrLoadResources[
  src={entries},% data in entries.bib
  ignore-fields={short,long},
  entry-type-aliases={foo=entry}
]

This treats the entry as though it had been defined as:

@entry{html,
  name = {HTML},
  description = {hypertext markup language}
}

whereas:

\GlsXtrLoadResources[
  src={entries},% data in entries.bib
  ignore-fields={name,description},
  entry-type-aliases={foo=abbreviation}
]

treats the entry as though it had been defined as:

@abbreviation{html,
  short = {HTML},
  long = {hypertext markup language}
}

unknown-entry-alias={value}

If this option is set, the ⟨value⟩ is used as the alias for any unknown entry types (after any aliases provided with entry-type-aliases have been applied). If the value is missing or empty, unknown entry types will be ignored with a warning.
5.3 General Options

action=⟨value⟩

This governs how the entries are written in the .glstex file. The ⟨value⟩ may be one of:

- define: define the entries;
- copy: copy the entries;
- define or copy: copy existing entries and define non-existing entries.

The default setting is action={define}, which writes the entry definition to the .glstex file using one of the commands described in section 6.1. Since the record package option automatically switches on the undefaction={warn} option, any attempt at defining an entry that’s already been defined will generate a warning rather than an error. The duplicate definition will be ignored. (The warnings can be found in the .log file since they are warnings produced by glossaries-extra not by bib2gls.)

For example, if you try:

\newglossary*{copies}{Copies}
\GlsXtrLoadResources[src={entries}]
\GlsXtrLoadResources[sort={use},type={copies},src={entries}]

you’ll find that the copies glossary is empty and there will be warnings in the .log file when the second resource file is loaded.

There are various ways of having the same entries in multiple glossaries. The simplest method is to use secondary, but another method is to use action={copy} which simply writes

\glsxtrcopytoglossary{⟨label⟩}{⟨type⟩}

instead of using one of the commands listed in section 6.1. This copies the entries rather than defining them, which means the entries must already have been defined. You can select entries that were selected in earlier resource sets with selection={selected before}.

The ⟨type⟩ is determined as follows:

- if the entry has the type field set, that’s used;
- if the entry is a tertiary and tertiary-type is set, that’s used;
- if the entry is a dual and dual-type is set, that’s used;
- otherwise the value of the type option is used.

If you’re not sure whether the entries may already be defined, you can use action={define or copy} which will use \ifglsentryexists in the resource file to determine whether to define or copy the entry.

Options that set or modify fields, such as category, group, save-locations, flatten or name-case-change, will be ignored if entries are copied. However the copy-action-group-field may be used to copy the group field (which may have been locally set by the sort
5.3 General Options

method) to another field. This ensures that the original group value from the entry definition in an earlier resource set won’t be overwritten (unless you set copy-action-group-field = {group}).

Remember that \glsxtrcopytoglossary simply copies the entry’s label to the glossary’s internal list. The only checks that bib2gls performs if action is not define is to ensure that the master or secondary options have not been used, since they’re incompatible, and that the type option is set, since it’s required as a fallback for any entries that don’t have the type field set. (There are too many options that alter field values to check them all and some may be used to alter the sorting.) The purpose of the copy action is simply to provide a duplicate list in a different order.

Remember that if you are using hyperref, you need to use target={false} in the optional argument of \printunsrtglossary for the glossary containing the copies to prevent duplicate hypertargets. Commands like \gls will link to the original entries. For example, in the preamble:

\newignoredglossary{copies}

\GlsXtrLoadResources[src={entries}]

\GlsXtrLoadResources[
  sort={use},
  action={copy},
  type={copies},
  src={entries}
]

and later in the document:

\printunsrtglossary[title={Glossary (Alphabetical)},style={indexgroup}]
\printunsrtglossary[type={copies},title={Glossary (Order of Use)},
  style={index},nogroupskip,% no grouping
  target={false}]

Note also the need to use nogroupskip and a non-group style for the duplicates since the group field will have been assigned in the first resource set if bib2gls was invoked with --group. The grouping is appropriate for alphabetical ordering but not for order of use.

If you want different grouping for the duplicates, you can specify the field name to use in which to store the group information using copy-action-group-field. Unlike secondary, you will need to redefine \glsxtrgroupfield to the relevant field before you display the glossary. The simplest way to do this is with the starred form of \printunsrtglossary. For example, if copy-action-group-field={dupgroup} is added to the options for the second resource set:

\printunsrtglossary*[type={copies},title={Duplicates},style={indexgroup}]
  {\renewcommand{\glsxtrgroupfield}{dupgroup}}
This just does:

\begingroup
\renewcommand{\glsxtrgroupfield}{dupgroup}\
\printunsrtglossary[type={copies},title={Duplicates},
    style={indexgroup}]\
\endgroup

\texttt{copy-to-glossary=\{list\}}

This option can selectively copy an entry to a glossary after it has been defined. If the supplied value \texttt{\{list\}} is empty, no copying is performed (except as a result of other options, such as \texttt{action} or \texttt{secondary}). If set, the \texttt{\{list\}} argument is a list of string concatenations with optional conditionals. Take care that constant strings are correctly delimited, as described below, to ensure that they are not mistaken for field labels.

The evaluation of the target glossary label for each entry is performed while the \texttt{.g1stex} file is being written (after sorting) so all field values should be available in any field reference. The \texttt{action} option is implemented first, so the selected entry will first either be defined or copied according to \texttt{action}. If the \texttt{copy-to-glossary} instruction is successful, the entry will then be copied to the target glossary using \texttt{\bibglscopytoglossary}.

The \texttt{copy-to-glossary} value should be a comma-separated list, where the syntax for each item in the list is in the form:

\texttt{\{element-list\} \{condition\}}

where \texttt{\{element-list\}} is a string concatenation (see section 5.1) and \texttt{\{condition\}} is a complex conditional (see section 5.2). For each \texttt{\{element-list\} \{condition\}} specification, if the condition evaluates to false or if the \texttt{\{element-list\}} evaluates to null then the copy instruction won’t be added. The fallback action for a missing field value is governed by the \texttt{copy-to-glossary-missing-field-action} setting. The result of the string concatenation (if not null) is the label of the target glossary.

You can have multiple copy instructions to copy an entry to multiple glossaries. The definition of \texttt{\bibglscopytoglossary} will ensure that an entry will only be copied to the designated glossary if it isn’t already in the glossary’s internal list and will silently do nothing if the glossary doesn’t exist.

Remember that constant strings need to be marked with braces or double-quote delimiters. For example, if you want to copy all entries to the index glossary then either do:

\texttt{copy-to-glossary=\{"index\"\}}

or

\texttt{copy-to-glossary=\{"index\}}

Note that the outer braces are stripped by the resource option parser, which first splits the \texttt{\{option\}={\{value\}}} list supplied via \texttt{\GlsXtrLoadResources} into \texttt{\{option\}} and \texttt{\{value\}} pairs.
and then parses each \langle option \rangle. So by the time that the copy-to-glossary option has its value parsed, the value has become "index" or \{index\}, respectively, in the above two examples.

Remember that the \langle value \rangle itself may be a comma-separated list. The outer grouping hides the inner list comma from the initial \langle option \rangle=\{\langle value \rangle \} split. For example, to copy all entries to the index and symbols glossaries:

\texttt{copy-to-glossary=\{"index", "symbols"\}}

or

\texttt{copy-to-glossary=\{\{index\}, \{symbols\}\}}

The following example will only copy entries to the index glossary if their actual entry type is index:

\texttt{copy-to-glossary=\{"index" \[ entrytype -> actual = "index" \] \}}

Alternatively, to copy aliased custom entry types \@person entries to a custom glossary \texttt{person} and \@place to a custom glossary \texttt{place}:

\texttt{copy-to-glossary=\{entrytype -> original \[ entrytype -> original =/person|place/ \] \}}

If the glossary types don’t conveniently match the entry type, the instructions can be split into a list. For example:

\texttt{copy-to-glossary=\{"abbreviations" \[ entrytype -> actual = "abbreviation" \],
"symbols" \[ entrytype -> actual = "symbol" \],
"numbers" \[ entrytype -> actual = "number" \] \}}

Each instruction in the list will be tried and the copy instruction will only be written if the condition evaluates to true and a non-null value is successfully returned.

\texttt{copy-to-glossary-missing-field-action=\langle value \rangle}

This option indicates what to do if a source field identified in copy-to-glossary is missing. The value may be one of:

- \texttt{skip}: return null;
- \texttt{fallback}: use the fallback for the missing field (see section 5.8), if one is available, otherwise return null (default);
- \texttt{empty}: treat the missing value as empty.

Returning null will result in the copy instruction being omitted.
5.4 Selection Options

src=⟨list⟩

This identifies the .bib files containing the entry definitions. The value should be a comma-separated list of the required .bib files. These may either be in the current working directory or in the directory given by the --dir switch or on \TeX’s path (in which case kpsewhich will be used to find them). The .bib extension may be omitted. Remember that if ⟨list⟩ contains multiple files it must be grouped to protect the comma from the ⟨options⟩ list.

For example:

\GlsXtrLoadResources[src={entries-terms,entries-symbols}]

indicates that bib2gls must read the files entries-terms.bib and entries-symbols.bib and create the file given by \jobname.glstex on the first instance or \jobname-⟨n⟩.glstex on subsequent use.

With \glsxtrresourcefile[(⟨options⟩)]{⟨filename⟩}, if the src option is omitted, the .bib file is assumed to be ⟨filename⟩.bib. For example:

\glsxtrresourcefile{entries-symbols}

indicates that bib2gls needs to read the file entries-symbols.bib, which contains the entry data, and create the file entries-symbols.glstex. If the .bib file is different or if you have multiple .bib files, you need to use the src option.

\GlsXtrLoadResources uses \jobname as the argument of \glsxtrresourcefile on the first instance, so:

\GlsXtrLoadResources[]

will assume src={\jobname}. Remember that subsequent uses of \GlsXtrLoadResources append a suffix, so in general it’s best to always supply src, except for small test cases with a single resource command.

With old \TeX kernels, if you have non-ASCII characters in the .bib filename but aren’t using X\TeX or Lua\TeX, then you will need to use \detokenize to prevent expansion when the information is written to the .aux file. Newer \TeX kernels have better support for UTF-8. Similarly for any special characters that need protecting (although it’s better not to use special characters in filenames). For example:

\documentclass{article}
\usepackage[T2A]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[russian]{babel}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[
  src={\detokenize{кириллица}}],% data in кириллица.bib
5.4 Selection Options

\begin{document}
\printunsrtglossary
\end{document}

\texttt{selection=}⟨value⟩

By default all entries that have records in the .aux file will be selected as well as all their dependent entries. The dependent entries that don’t have corresponding records on the first \LaTeX run, may need an additional build to ensure their location lists are updated.

Remember that on the first \LaTeX run the .glstex files don’t exist. This means that the entries aren’t defined at that point. The record package option additionally switches on the undefaction=\{warn\} option, which means that you’ll only get warnings rather than errors when you reference entries in the document. You can’t use \glsaddall with \bib2gls because the glossary lists are empty on the first run, so there’s nothing for \glsaddall to iterate over. Instead, if you want to add all defined entries, you need to instruct \bib2gls to do this with the selection option. The following values are allowed:

- \textbf{recorded and deps:} add all recorded entries and their dependencies (default).

- \textbf{recorded and deps and see:} as above but will also add unrecorded entries whose see, seealso or alias field refers to a recorded entry.

- \textbf{recorded and deps and see not also:} as above but will add unrecorded entries whose see or alias (but not seealso) field refers to a recorded entry.

- \textbf{recorded no deps:} add all recorded entries but not their dependencies. The dependencies include those referenced in the see or seealso field or fields identified by dependency-fields, parent entries and those found referenced with commands like \gls in the field values that are parsed by \bib2gls. With this setting, parents will be omitted unless they’ve been referenced in the document through commands like \gls. This setting won’t add any see or seealso lists to the location list. The given field will be set, so you can access the information, but there’s no guarantee that the cross-referenced entry will have been selected. The alias cross-reference will be added to the location list but you will need to ensure that the target is also selected (or use alias=\{omit\} to suppress it).

- \textbf{recorded and ancestors:} this is like the previous setting but parents are added even if they haven’t been referenced in the document. The other dependent entries are omitted if they haven’t been referenced in the document. The above notes regarding the cross-reference lists also applies.
5.4 Selection Options

- **deps but not recorded**: this first selects entries as though recorded and deps had been used, but after all ancestors and dependencies have been added it then removes all entries that have records. This means that you end up with only the unrecorded dependencies. (Recorded entries will need to be selected in a different resource set.)

- **ancestors but not recorded**: this first selects entries as though recorded and ancestors had been used, but after all ancestors have been added it then removes all entries that have records. This means that you end up with only the unrecorded ancestors. (Recorded entries will need to be selected in a different resource set.) See the sample-nested.tex example document.

- **selected before**: select any entries that have been selected in a previous resource set. This is intended for use with \texttt{action=\{copy\}} to copy entries to another glossary as an alternative to (or in addition to) the secondary option. Note that if you make any modifications to the fields (such as case-changing) the modification won’t be saved to the .glstex file. This option can’t be used in the first resource set.

- **all**: add all entries found in the .bib files supplied in the \texttt{src} option.

The \texttt{⟨value⟩} must be supplied.

For example, suppose the file entries.bib contains:

\begin{verbatim}
@index{run}
@index{sprint,see={run}}
@index{dash,see={sprint}}
\end{verbatim}

If the document only references the “run” entry (for example, using \texttt{\gls{run}}) then:

- If \texttt{selection=\{recorded and deps\}}, only the “run” entry is selected. The “run” entry has a record, so it’s selected, but it has no dependencies. Neither “sprint” nor “dash” have records, so they’re not selected.

- If \texttt{selection=\{recorded and deps and see\}}, the “run” and “sprint” entries are selected, but not the “dash” entry. The “run” entry is selected because it has a record. The “sprint” entry doesn’t have a record but its \texttt{see} field includes “run”, which does have a record, so “sprint” is also selected. The “dash” entry doesn’t have a record. Its \texttt{see} field references “sprint”. Although “sprint” has been selected, it doesn’t have any records, so “dash” isn’t selected.

The above is just an example. The circuitous redirection of “dash” to “sprint” to “run” is unhelpful to the reader and is best avoided (especially for an index where there are no accompanying descriptions and no location list for the intermediate “sprint”). A better method would be:

\begin{verbatim}
@index{run}
@index{sprint,see={run}}
@index{dash,see={run}}
\end{verbatim}
5.4 Selection Options

The `selection={recorded and deps and see}` in this case will select all three entries, and the document won’t send the reader on a long-winded detour.

Now suppose that the file `entries.bib` contains:

```latex
@entry{run,
  name = {run},
  description={move fast using legs}
}

@entry{sprint,
  name = {sprint},
  description={run at full speed over short distance},
  seealso={run}
}

@entry{dash,
  name = {dash},
  description={run in a great hurry},
  seealso={sprint}
}
```

and suppose the document only references “dash” (for example, with `\gls{dash}`), then with the default `selection={recorded and deps}` “dash” will be selected because it has a record, and “sprint” will be selected because “dash” requires it (for the cross-reference), and “run” will be selected because “sprint” requires it (for the cross-reference). In this case, neither “sprint” nor “run” have a location list but they do both provide additional information for the reader in their descriptions.

A better method here would be for each entry to have a cross-reference list that includes all related terms:

```latex
@entry{run,
  name = {run},
  description={move fast using legs},
  seealso={sprint,dash}
}

@entry{sprint,
  name = {sprint},
  description={run at full speed over short distance},
  seealso={run,dash}
}

@entry{dash,
  name = {dash},
  description={run in a great hurry},
  seealso={sprint}
}
```
5.4 Selection Options

```plaintext
seealso={sprint,run}
```

Now, whichever one is indexed in the document, the other two will automatically be selected.

```plaintext
match={key=value list}
```

It’s possible to filter the selection by matching field values. The value is required for this key but may be empty, which indicates that the setting is switched off, otherwise `key=value list` should be a `key={regexp}` list, where `key` is the name of a field or `id` for the entry’s label or `entrytype` for the `bib2gls` entry type (as in the part after `@` identifying the entry not the `type` field identifying the glossary label). If you’ve used `entry-type-aliases`, this refers to the target entry type not the original entry type specified in the `.bib` file.

The `regexp` part should be a regular expression conforming to Java’s Pattern class [5]. The pattern is anchored (`oo.*` matches `oops` but not `loops`) and `regexp` can’t be empty. Remember that `\TeX` will expand the option list as it writes the information to the `.aux` file so take care with special characters. For example, to match a literal period use `\string\.`, not `\.` (backslash dot).

If the field is missing its value it is assumed to be empty for the purposes of the pattern match even if it will be assigned a non-empty default value when the entry is defined. If the field is unrecognised by `bib2gls` any reference to it in `key=value list` will be ignored.

If a field is listed multiple times, the pattern for that field is concatenated using:

```plaintext
(?:(pattern-1))|(?:(pattern-2))
```

where `pattern-1` is the current pattern for that field and `pattern-2` is the new pattern. This means it performs a logical OR. For the non-duplicate fields the logical operator is given by `match-op`. For example:

```plaintext
match-op={and},
match={
    category=animals,
    topic=biology,
    category=vegetables
}
```

This will keep all the selected entries that satisfy:

- `category` matches `(?::animals)|(?:vegetables)`

  (the `category` is either `animals` or `vegetables`)

AND

- `topic` (custom key provided by user) is `biology`.  

136
and will discard any entries that don’t satisfy this condition. A message will be written to
the log file for each entry that’s discarded.

Patterns for unknown fields will be ignored. If the entire list consists of patterns for un-
known fields it will be treated as match=\{\}. That is, no filtering will be applied. In the above
example, the custom topic key must be provided before the first \GlsXtrLoadResources
with \glsaddkey or \glsaddstoragekey.

**match-op=⟨value⟩**

If the value of match contains more than one ⟨key⟩=⟨pattern⟩ element, the match-op deter-
mines whether to apply a logical AND or a logical OR. The ⟨value⟩ may be either and or or.
The default is match-op=\{and\}.

**not-match=⟨key=value list⟩**

If match=\{⟨key=value list⟩\} would cause an entry to be selected then not-match=\{⟨key=
value list⟩\} would cause that entry to be ignored. The value is required for this key but
may be empty, which indicates that the setting is switched off. If you have both match and
not-match in the same resource set, the last one listed takes precedence.

**match-action=⟨value⟩**

The default behaviour with match or not-match is to filter the selection. This may be
changed to append to the selection instead. The ⟨value⟩ may be one of:

- filter: (default) filter selection;
- add: append any matches (with match) or non-matches (with not-match) to the se-
  clection. This setting can’t be used with sort={use}.

For example, if I want to select all record entries and their dependencies, but I also want to
make sure that any entries with the category set to important are always selected regardless
of whether or not they have any records:

\GlsXtrLoadResources[
  src={entries},% data in entries.bib
  match-action={add},
  match={category=important}
]

**limit=⟨number⟩**

If ⟨number⟩ is greater than 0 then this will truncate the list of selected entries after sorting
to ⟨number⟩ (if the list size is greater than that value). The transcript will show the message:

Truncating according to limit=⟨number⟩
5.5 Hierarchical Options

When used with \texttt{shuffle}, this provides a means of randomly selecting at most $\langle \text{number} \rangle$ entries. The default setting is \texttt{limit=0} (no truncation). A negative value of $\langle \text{number} \rangle$ is not permitted.

If you have any dual entries, then the truncation will be applied to the combined list of primary and duals if \texttt{dual-sort=combine} otherwise each list will be truncated separately by $\langle \text{number} \rangle$, which results in a maximum of $2 \times \langle \text{number} \rangle$. Remember that tertiary entries are created when dual entries are defined in the .\texttt{glstex} file, so this will increase the total number of entries.

5.5 Hierarchical Options

Hierarchy is established by setting the \texttt{parent} field to the label of the parent entry. The parent and child entries are sorted together, but hierarchical comparators will place child entries after their corresponding parent.

The \texttt{glossaries} package provides \texttt{\ifglshasparent} to determine whether or not an entry has the \texttt{parent} field set. If also provides \texttt{\ifglshaschildren}, but this command is inefficient as it has to iterate over all entries to find an entry with the \texttt{parent} field set to the relevant label. It’s also non-trivial to determine which child entries have been included in the glossary with \texttt{makeindex} or \texttt{xindy}. \texttt{bib2gls} can provide this information with some of the options described in this section.

It’s also possible to flatten entries (that is, remove the hierarchical information) or just flatten lonely child entries.

\texttt{save-child-count=\langle boolean \rangle}

This is a boolean option. The default setting is \texttt{save-child-count={false}}. If \texttt{save-child-count={true}}, each entry will be assigned a field called \texttt{childcount} with the value equal to the number of child entries that have been selected. As from version 1.5, this option also creates the \texttt{childlist} field for entries that have children selected. This field is in etoolbox’s internal list format and can be iterated over using \texttt{\glsxtrfieldforlistloop}.

The assignment is done using \texttt{\GlsXtrSetField} so there’s no associated key. You can test if the field is set and non-zero using:

\begin{verbatim}
\GlsXtrIfHasNonZeroChildCount{\langle entry label \rangle}{\langle true \rangle}{\langle false \rangle}
\end{verbatim}

which is provided with glossaries-extra-bib2gls v1.31+. Within \texttt{\langle true \rangle}, you can access the actual value with \texttt{\glscurrentfieldvalue}. If \texttt{save-child-count={false}}, this command will do \texttt{\langle false \rangle} as the \texttt{childcount} field won’t be set.

For example, suppose entries.bib contains:

\begin{verbatim}
@index{birds}
@index{duck,parent={birds}}
@index{goose,plural={geese},parent={birds}}
@index{swan,parent={birds}}
\end{verbatim}
5.5 Hierarchical Options

@index{minerals}
@index{quartz,parent={minerals}}
@index{corundum,parent={minerals}}
@index{amethyst,parent={minerals}}
@index{gypsum,parent={minerals}}
@index{gold,parent={minerals}}

and the document contains:

\documentclass{article}
\usepackage[record,style={indexgroup}]{glossaries-extra}
\GlsXtrLoadResources[src={entries},save-child-count]
\begin{document}
\gls{duck} and \gls{goose}.
\gls{quartz}, \gls{corundum}, \gls{amethyst}.
\printunsrtglossaries
\end{document}

Then the .glstex file will contain:

\GlsXtrSetField{birds}{childcount}{2}
\GlsXtrSetField{duck}{childcount}{0}
\glsxtrfieldlistadd{birds}{childlist}{duck}
\GlsXtrSetField{goose}{childcount}{0}
\glsxtrfieldlistadd{birds}{childlist}{goose}
\GlsXtrSetField{minerals}{childcount}{3}
\GlsXtrSetField{amethyst}{childcount}{0}
\glsxtrfieldlistadd{minerals}{childlist}{amethyst}
\GlsXtrSetField{corundum}{childcount}{0}
\glsxtrfieldlistadd{minerals}{childlist}{corundum}
\GlsXtrSetField{quartz}{childcount}{0}
\glsxtrfieldlistadd{minerals}{childlist}{quartz}

Note that although birds has three children defined in the .bib file, only two have been selected, so the child count is set to 2. Similarly the minerals entry has five children defined in the .bib file, but only three have been selected, so the child count is 3.

The following uses the post-description hook to show the child count in parentheses:

\GlsXtrLoadResources[src={entries},category={general},save-child-count]
\glsdefpostdesc{general}{%
5.5 Hierarchical Options

\glsxtrifhasfield{childcount}{\glscurrententrylabel}%
{ (child count: \glscurrentfieldvalue.})%
{}%
}
\glsxtrifhasfield requires at least glossaries-extra v1.19. It’s slightly more efficient than \ifglshasfield provided by the base glossaries package, and it doesn’t complain if the entry or field don’t exist, but note that \glsxtrifhasfield implicitly scopes its content. Use the starred version to omit the grouping. With glossaries-extra v1.31+ you can perform a numerical test with \GlsXtrIfFieldNonZero or \GlsXtrIfFieldEqNum.

\texttt{save-sibling-count}=\langle boolean \rangle

This is a boolean option. The default setting is \texttt{save-sibling-count=\{false\}}. This is like \texttt{save-child-count} but saves the sibling count in \texttt{siblingcount} and the sibling list in \texttt{siblinglist}. As with the childlist, the sibling list is in etoolbox’s internal list format. The sibling information is only saved for entries that have a parent.

The advantage with \texttt{siblinglist} over accessing the parent’s \texttt{childlist} is that the entry itself is excluded from the list.

\texttt{save-root-ancestor}=\langle boolean \rangle

This is a boolean option. The default setting is \texttt{save-root-ancestor=\{false\}}. If true, the entry’s top-most ancestor will be saved in the entry’s \texttt{rootancestor} internal field. If the entry doesn’t have a parent (that it, the entry itself is the root) then the \texttt{rootancestor} field won’t be set.

\texttt{flatten}=\langle boolean \rangle

This is a boolean option. The default value is \texttt{flatten=\{false\}}. If \texttt{flatten=\{true\}}, the sorting will ignore hierarchy and the \texttt{parent} field will be omitted when writing the definitions to the \texttt{.glstex} file, but the parent entries will still be considered a dependent ancestor from the selection point of view.

Note the difference between this option and using \texttt{ignore-fields=\{parent\}} which will remove the dependency (unless a dependency is established through another field).

\texttt{flatten-lonely}=\langle value \rangle

This may take one of three values: false (default), presort and postsort. The value must be supplied.

Unlike the \texttt{flatten} option, which completely removes the hierarchy, the \texttt{flatten-lonely} option can be used to selectively alter the hierarchy. In this case only those entries that have a parent but have no siblings are considered. This option is affected by the \texttt{flatten-lonely-rule} setting. The conditions for moving a child up one hierarchical level are as follows:
5.5 Hierarchical Options

- The child must have a parent, and
- the child can’t have any selected siblings, and
- if flatten-lonely-rule={only unrecorded parents} then the parent can’t have a location list, where the location list includes records and \textit{see} or \textit{seealso} cross-references (for the other rules the parent may have a location list as long as it only has the one child selected).

If the child is selected for hierarchical adjustment, the parent will be removed if:

- The parent has no location list, and
- \texttt{flatten-lonely-rule} isn’t set to \texttt{no discard}.

The value of \texttt{flatten-lonely} determines whether the adjustment should be made before sorting (presort) or after sorting (postsort). To disable this function use \texttt{flatten-lonely} =\{false\}.

For example, suppose the file \texttt{entries.bib} contains:

\begin{verbatim}
@index{birds}
@index{duck, parent={birds}}
@index{goose, plural={geese}, parent={birds}}
@index{swan, parent={birds}}
@index{chicken, parent={birds}}

@index{vegetable}
@index{cabbage, parent={vegetable}}

@index{minerals}
@index{quartz, parent={minerals}}
@index{corundum, parent={minerals}}
@index{amethyst, parent={minerals}}
@index{gypsum, parent={minerals}}

@index{aardvark}
@index{bard}
@index{buzz}

@index{item}
@index{subitem, parent={item}}
@index{subsubitem, parent={subitem}}
\end{verbatim}

and suppose the document contains:
\begin{verbatim}
\documentclass{article}
\end{verbatim}
5.5 Hierarchical Options

\usepackage[record,style={indexgroup}]{glossaries-extra}
\GlsXtrLoadResources[src={entries.bib}]
\begin{document}
\gls{duck}.
\gls{quartz}, \gls{corundum}, \gls{amethyst}.
\gls{aardvark}, \gls{bard}, \gls{buzz}.
\gls{vegetable}, \gls{cabbage}.
\gls{subsubitem}.
\printunsrtglossaries
\end{document}

Although the duck entry has siblings in the entries.bib file, none of them have been recorded in the document, nor has the parent birds entry.

This document hasn’t used flatten-lonely, so the default flatten-lonely={false} is assumed. This results in the hierarchical structure:

A
aardvark 1

B
bard 1
birds
duck 1
buzz 1

I
item
subitem
subsubitem 1

M
minerals
amethyst 1
corundum 1
quartz 1
5.5 Hierarchical Options

V
vegetable 1
  cabbage 1
(The “1” in the above indicates the page number.) There are some entries here that look a little odd: duck, cabbage and subsubitem. In each case they are a lone child entry. It would look better if they could be compressed, but I don’t want to use the flatten option, as I still want to keep the mineral hierarchy.

If I now add flatten-lonely={postsort}:

\GlsXtrLoadResources[src={entries.bib},flatten-lonely={postsort}]

the hierarchy becomes:

A
aardvark 1

B
bard 1
birds, duck 1
buzz 1

I
item, subitem, subsubitem 1

M
minerals
  amethyst 1
  corundum 1
  quartz 1

V
vegetable 1
  cabbage 1

The name field of the duck entry has been set to:

name={\bibglslflattenedchildpostsort{birds}{duck}}
the text field has been set to:

\texttt{text={duck}}

the group field is copied over from the parent entry ("B"), and the parent field has been adjusted, moving duck up one hierarchical level. Finally, the former parent birds entry has been removed (the default flatten-lonely-rule={only unrecorded parents} is in effect).

The default definition of \texttt{\bibglsflattenedchildpostsort} formats its arguments so that they are separated by a comma and space ("birds, duck"). If the text field had been set in the original @index definition of duck, it wouldn’t have been altered. This adjustment ensures that in the document \texttt{\gls{duck}} still produces “duck” rather than “birds, duck”. (If the child and parent name fields are identical, the terms are considered homographs. See below for further details.)

The subsubitem entry has also been adjusted. This was done in a multi-stage process, starting with sub-items and then moving down the hierarchical levels:

- The subitem entry was adjusted, moving it from a sub-entry to a top-level entry. The name field was then modified to:

\texttt{name={\bibglsflattenedchildpostsort{item}{subitem}}}

This now means that the subsubitem entry is now a sub-entry (rather than a sub-sub-entry). The subitem entry now has no parent, but at this stage the subsubitem entry still has subitem as its parent.

- The subsubitem entry is then adjusted moving from a sub-entry to a top-level entry. The name field was then modified to:

\texttt{name=}
\%
\texttt{\bibglsflattenedchildpostsort}
\%
\%
\% name from former parent
\texttt{\bibglsflattenedchildpostsort{item}{subitem}}%
\%
\{subsubitem\} original name

The first argument of \texttt{\bibglsflattenedchildpostsort} is obtained from the name field of the entry’s former parent (which is removed from the child’s set of ancestors). This field value was changed in the previous step, and the change is reflected here. This means that the name for subitem will be displayed as “item, subitem” and the name for subsubitem will be displayed as “item, subitem, subsubitem”.

- The parent entries item and subitem are removed from the selection as they have no location lists.
Note that the cabbage sub-entry hasn’t been adjusted. It doesn’t have any siblings but its parent entry (vegetable) has a location list so it can’t be discarded. If I change the rule:

```
\GlsXtrLoadResources[src={entries.bib},
    flatten-lonely-rule={discard unrecorded},
    flatten-lonely={postsort}
]
```

then this will move the cabbage entry up a level but the original parent entry vegetable will remain:

```
A
aardvark 1

B
bard 1
birds, duck 1
buzz 1

I
item, subitem, subsubitem 1

M
minerals
    amethyst 1
    corundum 1
    quartz 1

V
vegetable 1
vegetable, cabbage 1
```

Remember that `flatten-lonely={postsort}` performs the adjustment after sorting. This means that the entries are still in the same relative location that they were in with the original `flatten-lonely={false}` setting. For example, duck remains in the B letter group before “buzz”.

With `flatten-lonely={presort}` the adjustments are made before the sorting is performed. For example, using:
\GlsXtrLoadResources[src={entries.bib},
flatten-lonely-rule={discard unrecorded},
flatten-lonely={presort}]

the hierarchical order is now:

A
aardvark 1

B
bard 1
buzz 1

C
cabbage 1

D
duck 1

M
minerals
  amethyst 1
  corundum 1
  quartz 1

S
subsubitem 1

V
vegetable 1

This method uses a different format for the modified \texttt{name} field. For example, the duck entry now has:

\texttt{name=\bibglsflattenedchildpresort\{duck\}\{birds\}}
The default definition of \bibglsflattenedchildpresort simply does the first argument and ignores the second. The sorting is then performed, but the interpreter recognises this command and can deduce that the sort value for this entry should be duck, so “duck” now ends up in the D letter group.

If you provide a definition of \bibglsflattenedchildpresort in the @preamble, it will be picked up by the interpreter. For example:

@preamble{"\providecommand{\bibglsflattenedchildpresort}[2]{#1 (#2)}"}

Note that the text field is only changed if not already set. This option may have unpredictable results for abbreviations as the name field (and sometimes the text field) is typically set by the abbreviation style. Remember that if the parent entry doesn’t have a location list and the rule isn’t set to no discard then the parent entry will be discarded after all relevant entries and their dependencies have been selected, so any cross-references within the parent entry (such as \gls occurring in the description) may end up being selected even if they wouldn’t be selected if the parent entry didn’t exist.

With both presort and postsort, if the parent name is the same as the child’s name then the child is considered a homograph and the child’s name is set to:

\bibglsflattenedhomograph\{⟨name⟩\}\{⟨parent label⟩\}

instead of the corresponding \bibglsflattenedchild…sort. This defaults to just ⟨name⟩.

\textbf{flatten-lonely-rule=⟨value⟩}

This option governs the rule used by flatten-lonely to determine which sub-entries (that have no siblings) to adjust and which parents to remove. The value may be one of the following, where ⟨condition⟩ is the condition provided by flatten-lonely-condition:

- only unrecorded parents: Only the sub-entries that have a parent without a location list (and have ⟨condition⟩ evaluate to true) will be altered. The parent entry will be removed from the selection if the child entry is adjusted. This value is the default setting.
- discard unrecorded: This setting will adjust all sub-entries that have no siblings (and have ⟨condition⟩ evaluate to true) regardless of whether or not the parent has a location list. Only the parent entries that don’t have a location list will be removed from the selection if the child entry is adjusted.
- no discard: This setting will adjust all sub-entries that don’t have siblings (and have ⟨condition⟩ evaluate to true) regardless of whether or not the parent has a location list. No entries will be discarded, so parent entries that don’t have a location list will still appear in the glossary.

In the above, the location list includes records and cross-references obtained from the see or seealso fields. See flatten-lonely for further details.
5.5 Hierarchical Options

**flattent-lonely-condition=(value)**

The value may either be empty, to indicate true (the default), or a complex condition using syntax described in section 5.2. After taking into account `flattent-lonely` and `flattent-lonely-rule`, this option determines whether or not the child entry will be adjusted. If the condition evaluates to false, the child entry won’t be adjusted.

For example, if both the parent entry and the child entry have long names, it may be better to keep their hierarchy. The following will only flatten lonely entries where both the child name and the parent name have less then 25 characters:

```
flattent-lonely={postsort},
flattent-lonely-condition={\LEN{parent -> name} < 25 & \LEN{name} < 25}
```

Alternatively, for a combined length of less than 50 characters:

```
flattent-lonely={postsort},
flattent-lonely-condition={\LEN{parent -> name + name} < 50}
```

This doesn’t include the number of characters taken up by the separator but the maximum value can be adjusted to allow for that, given a constant string separator.

**flattent-lonely-missing-field-action=(value)**

This option indicates what to do if a source field identified in `flattent-lonely-condition` is missing. The value may be one of:

- *skip*: return null;
- *fallback*: use the fallback for the missing field (see section 5.8), if one is available, otherwise return null (default);
- *empty*: treat the missing value as empty.

Returning null will result in the flatten lonely instruction being omitted.

**strip-missing-parents=(boolean)**

The glossaries package requires that all child entries must be defined after the parent entry. An error occurs otherwise, so `bib2gls` will omit the *parent* field if it can’t be found in the given resource set. However, when the default `strip-missing-parents={false}` is on, this omission only occurs while writing the definitions in the `.glstex` file (after selection and sorting).

Sorting is performed hierarchically and the *group* field is set accordingly for the top-level entries (but not for child entries), which means that an entry with a *parent* field will be treated by the sort method as a child entry. This can lead to a strange result, which `bib2gls` warns about:

```
Parent '(parent id)' not found for entry (child-id)
```
This is the default behaviour as it may simply be a result of a typing mistake in the parent field. If you actually want missing parents to be stripped before sorting (but after the selection process) then use strip-missing-parents={true}. If you want all parents stripped then use flatten or ignore-fields={parent} instead. As from version 1.4, if you want bib2gls to create the missing parents, then you can use missing-parents={create}.

missing-parents={⟨value⟩}

As an alternative to strip-missing-parents, as from version 1.4 you can now use missing-parents={⟨(value)⟩} where ⟨value⟩ may be one of:

- strip: this is equivalent to strip-missing-parents={true};
- warn: this is equivalent to the default strip-missing-parents={false};
- create: this will create a new @index entry with the missing parent’s label (after it’s been processed by options such as labelify) with the name obtained from the original value of the parent field (before being processed by options like labelify). If the child entry has the type field set, then the new parent entry will be given the same value. The category for the new parent entry can be assigned with missing-parent-category.

For example, consider the books.bib file which contains entries like:

```
@entry{ubik,
  name={Ubik},
  description={novel by Philip K. Dick},
  identifier={book},
  author={\sortmediacreator{Philip K.}{Dick}},
  year={1969}
}
```

then the field alias:

```
field-aliases={author=parent}
```

will treat:

```
author={\sortmediacreator{Philip K.}{Dick}},
```

as though it had been defined as:

```
parent={\sortmediacreator{Philip K.}{Dick}},
```

This can be converted into a label with the options:

```
labelify={parent},
labelify-replace={
  {{ \string\.}}{} 
}
```
5.5 Hierarchical Options

If the interpreter has been provided with the definition:

\providecommand*{\sortmediacreator}[2]{#2 #1}

then the parent field for the ubic entry will become DickPhilipK but the original value
is stored internally when missing-parents={create} is set so that it can be used as the
name if the parent needs to be created. Once all the entries have been processed, if ubic has
been selected but no entry can be found with the label DickPhilipK then a new entry will
be added as though it had been defined with:

@index{DickPhilipK,
 name={\sortmediacreator{Philip K.}{Dick}}
}

This is an alternative approach to the sample-authors.tex document from the examples
chapter.

missing-parent-category=⟨value⟩

If a missing parent entry is created through the use of missing-parents={create} then
the category field can be assigned to the new parent entry with this option. The ⟨value⟩
may be one of:

- same as child: the parent entry’s category field is set to the same value as the
  child’s (if set);
- same as base: the parent entry’s category is set to the base name of the .bib file
  that provided the child entry’s definition;
- no value or false: don’t set the category field;
- ⟨label⟩: the parent entry’s category field is set to ⟨label⟩ (which shouldn’t contain
  any special characters).

The default setting is missing-parent-category={no value}.

group-level=⟨value⟩

If letter group formation is enabled (see group, group-formation and --group) then the
default behaviour is to only assign the group label for top-level entries. This option allows
the group label to be assigned to sub-entries if sub-groups are required. The value may be
one of the following:

- ⟨n⟩: only assign the group for level ⟨n⟩ entries;
- >⟨n⟩: only assign the group for entries with a level greater than ⟨n⟩;
- >=⟨n⟩: only assign the group for entries with a level greater than or equal to ⟨n⟩;
5.5 Hierarchical Options

- `<\langle n \rangle>`: only assign the group for entries with a level less than \langle n \rangle;
- `\langle n \rangle \leq <\langle n \rangle>`: only assign the group for entries with a level less than or equal to \langle n \rangle;
- `all`: equivalent to `group-level={>=0}`.

The default setting is `group-level={0}`. If no value is provided, `group-level={all}` is assumed. The hierarchical levels start at 0 (top-level entry). For any value other than `group-level={0}`, the parent entry label will be included in the group label.

The hierarchical group titles are formatted according to `\bibglshiersubgrouptitle`. If the group title would usually be set with the command `\bibglsset…group` for top-level entries then the hierarchical group title would be set with the analogous `\bibglset…group` command. For example, letter groups are normally set with `\bibglsetlettergrouptitle` but hierarchical letter groups are set with `\bibglsetlettergrouptitlehier`.

If the `--no-group` setting is on then this option has no effect.

Any value other than the default `group-level={0}` requires `glossaries-extra v1.49+`, which provides `\glssubgroupheading`.

Sub-groups are implemented by the glossary style command:

\begin{verbatim}
\glssubgroupheading{\langle previous level \rangle}{\langle level \rangle}{\langle parent-label \rangle}{\langle group-label \rangle}
\end{verbatim}

The glossaries-extra package automatically implements:

\begin{verbatim}
\renewcommand*{\glssubgroupheading}{\glsgroupheading{#4}}
\end{verbatim}

whenever a style is set, so that if the style doesn’t provide a definition for this command, it will behave like `\glsgroupheading`.

`merge-small-groups=\langle n \rangle`

Merges consecutive small groups that have less than \langle n \rangle entries. The default is `merge-small-groups={0}`, which switches off this action. If \langle n \rangle is omitted, `merge-small-groups={1}` is assumed.

This setting only has an effect if group formation is enabled. If hierarchical sub-groups are enabled (`group-level`) then merging is only performed on consecutive small groups within the same hierarchical level. Any child entries that aren’t in their own sub-group are included in the higher level group count.

For example, suppose you have a large number of entries in most of the letter groups:

```
@index{aardvark}
@index{ant}
@index{alligator}
@index{ape}
% etc
```
but you only have one entry in each of the “X”, “Y” and “Z” groups:

@index{xylem}
@index{yak}
@index{zebra}

then you may prefer to merge these entries into a single group:

\GlsXtrLoadResources[merge-small-groups]

The title of this merged group is obtained from \bibglsmergedgrouptitle (or \bibgls-mergedgrouptitle if hierarchical groups have been enabled with group-level). For the above example, the merged letter group would have the title “X, Y, Z”. If there are more than three groups then the middle group titles are replace with an ellipsis. For example, if there is also only one entry in the “W” letter group, then the merged title would be “W,..., Z”.

The small groups must be consecutive (there is no group between them) and on the same hierarchical level in order to be merged. In the above example, if the yak entry isn’t selected so that there is no “Y” letter group, then the “X” and “Z” groups can be merged (with the merged title “X, Z”). If, on the other hand, extra entries occur in the “Y” letter group, so that it is larger than the value of merge-small-groups, then “X” and “Z” can no longer be merged.

5.6 Master Documents

Suppose you have two documents mybook.tex and myarticle.tex that share a common glossary that’s shown in mybook.pdf but not in myarticle.pdf. Furthermore, you’d like to use hyperref and be able to click on a term in myarticle.pdf and be taken to the relevant page in mybook.pdf where the term is listed in the glossary.

This can be achieved with the targeturl and targetname category attributes. For example, without bib2gls the file mybook.tex might look like:

\documentclass{book}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}

\makeglossaries

\newglossaryentry{sample}{name={sample},description={an example}}

\begin{document}
\chapter{Example}
\gls{sample}.

\printglossaries
\end{document}
The other document myarticle.tex might look like:

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

\newignoredglossary*{external}
\glssetcategoryattribute{external}{targeturl}{mybook.pdf}
\glssetcategoryattribute{external}{targetname}{\glolinkprefix\glslabel}

\newglossaryentry{sample}{type=external,category=external,
      name={sample},description={an example}}

\begin{document}
\gls{sample}.
\end{document}

In this case the main glossary isn’t used, but the category attributes allow a mixture of internal and external references, so the main glossary could be used for the internal references. (In which case, \texttt{makeglossaries} and \texttt{printglossaries} would need to be added back to myarticle.tex.)

Note that both documents had to define the common terms. The above documents can be rewritten to work with \texttt{bib2gls}. First a \texttt{.bib} file needs to be created:

\begin{verbatim}
@entry{sample,
    name={sample},
    description={an example}}
\end{verbatim}

Assuming this file is called myentries.bib, then mybook.tex can be changed to:

\documentclass{book}
\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[src={myentries}]

\begin{document}
\chapter{Example}
\gls{sample}.
\printunsrtglossaries
\end{document}

and myarticle.tex can be changed to:
Most of the options related to sorting and the glossary format are unneeded here since the glossary isn’t being displayed. This may be sufficient for your needs, but it may be that the book has changed various settings that have been written to mybook.glstex but aren’t present in the .bib file (such as short-case-change={uc}). In this case, you could just remember to copy over the settings from mybook.tex to myarticle.tex, but another possibility is to simply make myarticle.tex input mybook.glstex instead of using \GlsXtrLoadResources. This can work but it’s not so convenient to set the label prefix, the type and the category. The master option allows this, but it has limitations (see below), so in complex cases (in particular different label prefixes combined with hierarchical entries or cross-references) you’ll have to use the method shown in the example code above.

\texttt{master=⟨name⟩}

This option will disable most of the options that relate to parsing and processing data contained in .bib files (since this option doesn’t actually read any .bib files). It also can’t be used with action={copy} or action={define or copy}. A value of false will switch off this setting (the default).

The use of master isn’t always suitable. In particular if any of the terms cross-reference each other, such as through the see orseealso field or the parent field or using commands like \gls in any of the other fields when the labels have been assigned prefixes. In this case you will need to use the method described in the example above.

The ⟨name⟩ is the name of the .aux file for the master document without the extension (in this case, mybook). It needs to be relative to the document referencing it or an absolute path using forward slashes as the directory divider. Remember that if it’s a relative path, the PDF files (mybook.pdf and myarticle.pdf) will also need to be located in the same relative
When \texttt{bib2gls} detects the \texttt{master} option, it won’t search for entries in any \texttt{.bib} files (for that particular resource set) but will create a \texttt{.glstex} file that inputs the master document’s \texttt{.glstex} files, but it will additionally temporarily adjust the internal commands used to define entries so that the prefix given by \texttt{label-prefix}, the glossary type and the category type are all automatically inserted. If the \texttt{type} or \texttt{category} options haven’t been used, the corresponding value will default to \texttt{master}. The \texttt{targeturl} and \texttt{targetname} category attributes will automatically be set, and the glossary type will be provided using \texttt{\provideignoredglossary*\{⟨\texttt{type}⟩\}} (even if \texttt{--no-provide-glossaries} is set).

The above \texttt{myarticle.tex} can be changed to:

\begin{verbatim}
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[label-prefix={book.},master={mybook}]

\begin{document}
\gls{book.sample}.
\end{document}
\end{verbatim}

There are some settings from the master document that you still need to repeat in the other document. These include the label prefixes set when the master document loaded the resource files, and any settings in the master document that relate to the master document’s entries.

For example, if the master document loaded a resource file with \texttt{label-prefix={term.}} then you also need this prefix when you reference the entries in the dependent document in addition to the \texttt{label-prefix} for the dependent document. Suppose \texttt{mybook.tex} loads the resources using:

\begin{verbatim}
\GlsXtrLoadResources[src={myentries},label-prefix={term.}]
\end{verbatim}

and \texttt{myarticle.tex} loads the resources using:

\begin{verbatim}
\GlsXtrLoadResources[label-prefix={book.},master={mybook}]
\end{verbatim}

Then the entries referenced in \texttt{myarticle.tex} need to use the prefix \texttt{book.term.} as in:

\begin{verbatim}
This is a \gls{book.term.sample} term.
\end{verbatim}

Remember that the category labels will need adjusting to reflect the change in category label in the dependent document.

For example, if \texttt{mybook.tex} included:

\begin{verbatim}
\setabbreviationstyle{long-short-sc}
\end{verbatim}

then \texttt{myarticle.tex} will need:

\begin{verbatim}
\end{verbatim}

155
5.7 Field and Label Options

\setabbreviationstyle[master]{long-short-sc}

(change master to \textit{value} if you have used category={\textit{value}}). You can, of course, choose a different abbreviation style for the dependent document, but the category in the optional argument needs to be correct.

\texttt{master-resources=}\langle\textit{list}\rangle

If the master document has multiple resource files then by default all the master document’s .glstex files will be input. If you don’t want them all you can use master-resources to specify only those files that should be included. The value \textit{\langle list\rangle} is a comma-separated list of names, where each name corresponds to the final argument of \texttt{\glsxtrresourcefile}. Remember that \texttt{\GlsXtrLoadResources} is just a shortcut for \texttt{\glsxtrresourcefile} that bases the name on \texttt{\jobname}. (Note that, as with the argument of \texttt{\glsxtrresourcefile}, the .glstex extension should not be included in \textit{\langle list\rangle}.) The file \texttt{\jobname.glstex} is considered the primary resource file and the files \texttt{\jobname-\langle n\rangle.glstex} (starting with \textit{\langle n\rangle} equal to 1) are considered the supplementary resource files.

For example, to just select the first and third of the supplementary resource files (omitting the primary mybook.glstex):

\begin{verbatim}
\GlsXtrLoadResources[
    master={mybook},
    master-resources={mybook-1,mybook-3}
]
\end{verbatim}

5.7 Field and Label Options

The options in this section may be used to set or adjust field values or labels. Some field values are expected to be labels (such as \texttt{group}). These labels must not contain special characters or commands, but it’s possible to convert a field value into a valid label using options such as \texttt{labelify}.

Label Options

\texttt{interpret-label-fields=}\langle\textit{boolean}\rangle

This is a boolean option that determines whether or not the fields that may only contain labels should have their values interpreted (\texttt{parent, category, type, group, seealso} and \texttt{alias}). Although this option interprets commands within those fields, it doesn’t strip any characters that can’t be used within a label. The \texttt{see} field isn’t included as it may optionally start with \texttt{\langle tag\rangle} where \texttt{\langle tag\rangle} may legitimately contain \LaTeX{} code that shouldn’t be interpreted.

The default setting is \texttt{interpret-label-fields=}\langle\textit{false}\rangle. Note that if this setting is on, cross-resource references aren’t permitted. This setting has no effect if the interpreter has been disabled.
5.7 Field and Label Options

Related settings are `labelify` and `labelify-list` which can be used to strip content that can’t be used in labels and may be used more generally for other fields. The `labelify` and `labelify-list` options are performed before `interpret-label-fields`.

`labelify=(list)`

This option should take a comma-separated list of recognised field names as the value. (If a field is present in both `labelify` and `labelify-list`, then `labelify-list` takes precedence.) Note that if this setting is on, cross-resource references aren’t permitted. The value is required for this key but may be empty, which indicates an empty set of fields (that is, the setting is switched off).

Each listed field will be converted into a string suitable for use as a label. (Not necessarily a glossary entry label, but any label that may be used in the construction of a control sequence name.)

The conversion is performed in the following order:

1. If the interpreter is on and the field value contains any of the characters \ (backslash), { (begin group), } (end group), ~ (non-breakable space) or $ (maths shift), then the value is interpreted.

2. Any substitutions that have been specified with `labelify-replace` are performed.

3. All characters that aren’t alphanumeric or the space character or any of the following punctuation characters . (full stop), - (hyphen), + (plus), : (colon), ; (semi-colon), | (pipe), / (forward slash), ! (exclamation mark), ? (question mark), * (asterisk), < (less than), > (greater than), ` (backtick), ' (apostrophe) or @ (at-sign) are stripped. If you want to retain commas, use `labelify-list` instead. If you want to strip any of the allowed punctuation, use `labelify-replace` to remove the unwanted characters. (Remember that babel can make some of these punctuation characters active, in which case they need to be stripped.)

4. If `bib2gls` doesn’t allow non-ASCII characters in labels, the value is then decomposed and all non-ASCII characters are removed. UTF-8 support is automatic if `bib2gls` detects `fontspec` in the document’s transcript file, otherwise UTF-8 in labels will only be supported if `bib2gls` detects that the versions of `glossaries` and `glossaries-extra` are new enough to support it. To ensure better support for UTF-8 with pdfLaTeX, make sure you have a recent TeX distribution and up-to-date versions of `glossaries` and `glossaries-extra`.

For example, suppose the `.bib` file contains:

```latex
@index{sample, 
  name={\AA ngstr"om, \O stergaard, d'Arcy, and Fotheringay-Smythe} }
```

Then:
5.7 Field and Label Options

\GlsXtrLoadResources[
    src={entries},% data in entries.bib
    labelify={name}
]

will convert the name field into:

Angstrom stergaard d'Arcy and Fotheringay-Smythe

if bib2gls doesn't support non-ASCII characters in labels otherwise it will be:

Ångström Østergaard d'Arcy and Fotheringay-Smythe

Note that Ø is considered an unmodified letter and so can't be decomposed into a basic Latin
letter with a combining diacritic. It's therefore removed completely from the ASCII label
version. Whereas Å can be decomposed into “A” followed by the “combining ring above”
character and ö can be decomposed into “o” followed by the “combining diaresis” character.
You can use labelify-replace to replace non-ASCII characters into the closest match.
Alternatively, switch to using XƎL or LuaLƎX.

You can use this option with replicate-fields if you need to retain the original:

\GlsXtrLoadResources[
    src={entries},% data in entries.bib
    replicate-fields={name={user1}},
    labelify={user1}
]

labelify-list={list}

This option is like labelify but it retains commas, as it’s designed for fields that should
be converted into a comma-separated list of labels. Any empty elements are removed. For
example, with the .bib entry from above:

\GlsXtrLoadResources[
    src={entries},% data in entries.bib
    replicate-fields={name={user1}},
    labelify-list={user1}
]

will convert the user1 field into:

Angstrom, stergaard, d'Arcy, and Fotheringay-Smythe

or:

Ångström, Østergaard, d'Arcy, and Fotheringay-Smythe

depending on whether or not UTF-8 labels are supported.

158
5.7 Field and Label Options

\texttt{labelify-replace=\langle\text{list}\rangle}

This option takes a comma-separated list as a value with each element in the list in the form \{\langle\text{regex}\rangle\}{\langle\text{replacement}\rangle}\ where \langle\text{regex}\rangle\ is a regular expression (that conforms to Java’s Pattern class [5]) and \langle\text{replacement}\rangle\ is the replacement text. The value is required for this key but may be empty, which indicates that the setting is switched off.

Remember that the argument of \texttt{\GlsXtrLoadResources} is expanded when written to the .\texttt{aux} file so take care to protect any special characters. For example, to match a literal full stop use \texttt{\string\}. rather than just \texttt{.} (backslash dot).

In the \langle\text{replacement}\rangle\ part, you can use \texttt{\glscapturedgroup\langle n\rangle} to reference a captured sub-sequence. For example:

\texttt{labelify-replace=\{\{[A-Z]\}\texttt{\string\}.\}{\glscapturedgroup1}}

This removes any full stop that follows any of the characters A,...,Z. Alternatively, you can just use \texttt{\string\$} instead of \texttt{\glscapturedgroup}. If you want a literal dollar character, you need to use \texttt{\glshex24} (or \texttt{\string\u24}). This isn’t recommended for labels (since special characters are automatically stripped), but \texttt{sort-replace} follows the same rules as \texttt{labelify-replace}, and it may be needed for that.

\textbf{You can’t use the CMG quark (which expands to the CMG identifier in a string concatenation) to identify the captured group in this context, as the replacement text needs to use the correct regular expression syntax.}

Both \texttt{labelify} and \texttt{labelify-list} use the \texttt{labelify-replace} setting to perform substitutions. For example, to replace the sub-string “ and ” (including spaces) with a comma:

\begin{verbatim}
\GlsXtrLoadResources[ 
  src={entries}, % data in entries.bib 
  replicate-fields={name={user1}}, 
  labelify-replace={\{ \texttt{and} \}\{,\}}, 
  labelify-list={user1} ]
\end{verbatim}

The earlier example will now end up as:

Angstrom, stergaard, d'Arcy,Fotheringay-Smythe

or:

Ångström, Østergaard, d'Arcy,Fotheringay-Smythe

depending on whether or not UTF-8 labels are supported.

Note that this produces the same result regardless of whether or not the Oxford comma is present as ,\texttt{and}, would first be converted to , , and then the empty element is removed resulting in a single comma.

You can have more than one replacement:
This additionally removes the space, apostrophe and hyphen characters (second substitution) and replaces “Ø” (0x00D8) with “O” (third substitution) so the string now ends up as:

Angstrom, Ostergaard, dArcy, FotheringaySmythe

or:

Ångström, Ostergaard, dArcy, FotheringaySmythe

depending on whether or not UTF-8 labels are supported.

The label-prefix option prepends \textit{(tag)} to each entry’s label. This \textit{(tag)} will also be inserted in front of any cross-references, unless they start with \textit{dual}. or \textit{tertiary}. or \textit{ext}\textsubscript{\textit{n}}. (where \textit{n} is an integer). Use \textit{dual-prefix} to change the dual label prefixes and \textit{ext-prefixes} to change the external label prefixes.

If you set \textit{label-prefix} and you define commands with \texttt{\glsxtrnewglslike}, then any of those commands found in entry fields won’t have the \textit{label-prefix} inserted if the prefix provided with the command starts with the prefix given in \textit{label-prefix}. (This doesn’t apply to other prefix options, such as \textit{dual-prefix}, so take care if you have a mixture of prefix options and prefixes identified with \texttt{\glsxtrnewglslike}).

As from version 1.8, the primary label prefix is identified in the \texttt{.glistex} file with:

\begin{verbatim}
\bibglsprimaryprefixlabel{\textit{prefix}}
\end{verbatim}

For example, if the \texttt{.bib} file contains:

\begin{verbatim}
@entry{bird, 
   name={bird},
   description = {feathered animal, such as a \texttt{\gls{duck}} or \texttt{\gls{goose}}}
}

@entry{waterfowl, 
   name={waterfowl},
   description={Any \texttt{\gls{bird}} that lives in or about water},
\end{verbatim}

\texttt{\GlsXtrLoadResources[}
\begin{verbatim}
src={entries},% data in entries.bib
replicate-fields={name={user1}},
labelify-replace={
   { and },{},% first substitution
   {[ \texttt{\string\-}]{},% second substitution
   \texttt{\gls{00D8}}\texttt{O}{% third substitution
   },
   labelify-list={user1}
}\]
\end{verbatim}
\begin{verbatim}
This additionally removes the space, apostrophe and hyphen characters (second substitution) and replaces “Ø” (0x00D8) with “O” (third substitution) so the string now ends up as:

Angstrom, Ostergaard, dArcy, FotheringaySmythe

or:

Ångström, Ostergaard, dArcy, FotheringaySmythe

depending on whether or not UTF-8 labels are supported.

\texttt{\glsxtrnewglslike}
\end{verbatim}

\begin{verbatim}
\end{verbatim}
Then if this `.bib` file is loaded with `label-prefix={gls.}` it’s as though the entries had been defined as:

```latex
@entry{gls.bird,  
  name={bird},  
  description = {feathered animal, such as a \gls{gls.duck} or \gls{gls.goose}}  
}
```

```latex
@entry{gls.waterfowl,  
  name={waterfowl},  
  description={Any \gls{gls.bird} that lives in or about water},  
  see={[[see also]{gls.duck,gls.goose}}}  
}
```

```latex
@index{gls.duck,name={duck}}  
@index{gls.goose,name={goose},plural="geese"}  
```

Remember to use this prefix when you reference the terms in the document with commands like `\gls`.

**duplicate-label-suffix=(value)**

The glossaries package doesn’t permit entries with duplicate labels (even if they’re in different glossaries). If you have multiple resource sets and an entry that’s selected in one resource set is also selected in another, by default, `bib2gls` will issue a warning, but it will still write the entry definition to the `.glstex` file, which means you’ll also get a warning from `glossaries-extra` and the duplicate definition will be ignored, but associated internal fields set with commands like `\GlsXtrSetField` may still be set.

If you actually want the duplicate, you need to specify a suffix with `duplicate-label-suffix=(value)`. This suffix is only set just before writing the entry definition to the `.glstex` file, so it doesn’t affect selection criteria nor can label substitutions be performed in any cross-references. Options such as `set-widest` that reference entry labels are incompatible as they will use the unsuffixed label.

The actual suffix is formed from `(value)<n>` where `<n>` is an integer that’s incremented in the event of multiple duplicates. For example, `duplicate-label-suffix={.copy}` will change the label to `<id>.copy1` for the first duplicate of the entry whose label is `<id>`, and `<id>.copy2` for the second duplicate, etc.
5.7 Field and Label Options

**record-label-prefix={tag}**

If set, this option will cause bib2gls to pretend that each record label starts with \langle tag \rangle, if it doesn’t already. For example, suppose the records in the .aux file are:

\glxtr@record{bird}{}{page}{glsnumberformat}{1}
\glxtr@record{waterfowl}{}{page}{glsnumberformat}{1}
\glxtr@record{idx.duck}{}{page}{glsnumberformat}{1}
\glxtr@record{idx.goose}{}{page}{glsnumberformat}{1}

The use of **record-label-prefix={idx.}** makes bib2gls act as though the records were given as:

\glxtr@record{idx.bird}{}{page}{glsnumberformat}{1}
\glxtr@record{idx.waterfowl}{}{page}{glsnumberformat}{1}
\glxtr@record{idx.duck}{}{page}{glsnumberformat}{1}
\glxtr@record{idx.goose}{}{page}{glsnumberformat}{1}

**cs-label-prefix={tag}**

If you have commands such as \texttt{\gls{label}} or \texttt{\glstext{label}} in field values (in situations where nested link text won’t cause a problem) the \langle label \rangle will be converted as follows:

- if \langle label \rangle starts with dual. then dual. will be replaced by the dual-prefix value;
- if \langle label \rangle starts with tertiary. then tertiary. will be replaced by the tertiary-prefix value;
- if \langle label \rangle starts with ext \langle n \rangle. then ext \langle n \rangle. will be replaced by the corresponding ext-prefixes setting (if cross-resource reference mode is enabled, see section 1.4);
- if \langle label \rangle doesn’t start with one of the above recognised prefixes then, if cs-label-prefix has been used the supplied value will be inserted otherwise the label-prefix setting will be inserted.

For example, given:

```latex
@entry{bird,
  name={bird},
  description = {feathered animal, such as a \gls{duck} or \gls{goose}}
}
```

then if label-prefix={idx.} is set but cs-label-prefix isn’t included in the resource option list this will convert the description field to:

```
description = {feathered animal, such as a \gls{idx.duck} or \gls{idx.goose}}
```

However with cs-label-prefix={gls.} the description field will be converted to:
5.7 Field and Label Options

description = \{feathered animal, such as a \texttt{\textbackslash gls}\{\texttt{\textbackslash gls.duck}\} or \texttt{\textbackslash gls}\{\texttt{\textbackslash gls.goose}\}\}

regardless of the label-prefix setting. Whereas if the original entry definition is:

\begin{verbatim}
@entry{bird,
    name={bird},
    description = \{feathered animal, such as a \texttt{\textbackslash gls}\{dual.duck\} or \texttt{\textbackslash gls}\{dual.goose\}\}
}
\end{verbatim}

then dual. will be replaced by the value of the dual-prefix option regardless of the cs-label-prefix setting.

The cs-label-prefix setting doesn’t affect labels in the fields that have an entry label or label list as the value (parent, alias, see and seealso).

\texttt{ext-prefixes=\langle list\rangle}

Any cross-references in the .bib file that start with \texttt{ext\langle n\rangle}. (where \langle n\rangle is a positive integer) will be substituted with the \langle n\rangle th tag listed in the comma-separated \langle list\rangle. If there aren’t that many items in the list, the ext\langle n\rangle. will simply be removed. The default setting is an empty list, which will strip all ext\langle n\rangle. prefixes. Remember that cross-resource reference mode needs to be enabled for this option to work (see section 1.4).

As from version 1.8, the external label prefixes are identified in the .glstex file with:

\begin{verbatim}
\bibglsextexternalprefixlabel{\langle n\rangle}{\langle prefix\rangle}
\end{verbatim}

For example, suppose the file entries-terms.bib contains:

\begin{verbatim}
@entry{set,
    name={set},
    description={collection of values, denoted \texttt{\textbackslash gls}\{ext1.set\}}
}
\end{verbatim}

and the file entries-symbols.bib contains:

\begin{verbatim}
@symbol{set,
    name={\texttt{\textbackslash ensuremath}{\texttt{\textbackslash mathcal\{S\}}}},
    description={a \texttt{\textbackslash gls}\{ext1.set\}}
}
\end{verbatim}

These files both contain an entry with the label set but the description field includes \texttt{\textbackslash gls}\{ext1.set\} which is referencing the entry from the other file. These two files can be loaded without conflict using:

\begin{verbatim}
\usepackage[record,symbols]{glossaries-extra}
\GlssXtrLoadResources[src=\{entries-terms\},
\end{verbatim}
label-prefix={gls.},
ext-prefixes={sym.}
]

\GlsXtrLoadResources[src={entries-symbols},
type={symbols},
label-prefix={sym.},
ext-prefixes={gls.}
]

Now the set entry from entries-terms.bib will be defined with the label gls.set and
the description will be:

\textbf{collection of values, denoted \textbackslash gls\{sym.set\}}

The set entry from entries-symbols.bib will be defined with the label sym.set and the
description will be:

\textbf{a \textbackslash gls\{gls.set\}}

Note that in this case the .bib files have to be loaded as two separate resources. They
can’t be combined into a single src list as the labels aren’t unique.

If you want to allow the flexibility to choose between loading them together or separately,
you’ll have to give them unique labels. For example, entries-terms.bib could contain:

@entry{set,
    name={set},
    description={collection of values, denoted \textbackslash gls\{ext1.S\}}
}

and entries-symbols.bib could contain:

@symbol{S,
    name={\textbackslash ensuremath\{\textbackslash mathcal\{S\}\}},
    description={a \textbackslash gls\{ext1.set\}}
}

Now they can be combined with:

\GlsXtrLoadResources[src={entries-terms,entries-symbols}]

which will simply strip the ext1. prefix from the cross-references. Alternatively:

\GlsXtrLoadResources[src={entries-terms,entries-symbols},
    label-prefix={gls.},
    ext-prefixes={gls.}
]

which will insert the supplied label-prefix at the start of the labels in the entry definitions
and will replace the ext1. prefix with gls. in the cross-references.
5.7 Field and Label Options

`prefix-only-existing=⟨boolean⟩`

This is a boolean option. It’s possible that a prefix can end up being inserted when there’s no entry in the current resource set that matches the prefixed label. If this option is set then the prefix won’t be added if there’s no matching entry. The default setting is `prefix-only-existing={false}`.

`save-original-id=⟨value⟩`

The ⟨value⟩ may be the keywords `false` or `true` or the name of a field in which to store the entry’s original label (as given in the .bib file). The default setting is `save-original-id={false}`. If ⟨value⟩ is omitted or is the keyword `true`, then `originalid` is assumed.

If ⟨value⟩ has an associated key in `\newglossaryentry` (for example, one provided with `\glsaddstoragekey`) it will be set after the field aliases, otherwise (for example, `original-id`) it will simply be added to the .glstex file using `\GlsXtrSetField` after the entry definition (which means the field can’t be referenced in other resource options). This setting is governed by `save-original-id-action`.

`save-original-id-action=⟨value⟩`

This option determines whether or not `save-original-id` should save the original entry label. No action is performed when `save-original-id={false}` otherwise the action is determined by ⟨value⟩ which may be one of the following keywords:

- **always**: always save the original label (default);
- **no override**: don’t override a field that’s already been set;
- **changed override or changed or diff**: only save the original label if it’s different from the final label;
- **changed no override**: only save the original label if it’s different from the final label and the specified field hasn’t been set.

The "no override" options make no difference if the given field has no associated key in `\newglossaryentry` (such as `originalid`). For known fields, bear in mind that the field will be set after field aliasing but before other options, such as `ignore-fields`.

`save-definition-index=⟨boolean⟩`

This is a boolean option. If the value is omitted `true` is assumed. The default setting is `save-definition-index={false}`.

This setting will save the definition index that’s used by `identical-sort-action={def}` to determine the order of definition in the special internal field `definitionindex`. This field is assigned when the entry is first created and can be referenced with `\bibglsdefinitionindex`. You can reference this field with certain resource options, such as `format-integer-fields`, but you must place the `save-definition-index` resource option first.
Note that (unless you need to maintain hierarchy) if you want to order all entries by definition, it’s better to use `sort={none}`, which doesn’t perform any sorting, so the order will be by definition.

**save-use-index=(boolean)**

This is a boolean option. If the value is omitted `true` is assumed. The default setting is `save-use-index={false}`.

This setting will save the order of use index that’s used by `identical-sort-action={use}` in the special internal field `useindex`. This field is assigned when the entry picks up its first record and can be referenced with `\bibglsuseindex`. You can’t reference this field in resource options such as `format-integer-fields`.

Entries that don’t have records won’t have this field set. The order of use corresponds to the first time the entry is recorded in the document.

Note that (unless you need to maintain hierarchy) if you want to order all entries by use, it’s better to use `sort={use}`, which doesn’t perform any sorting.

**dependency-fields=(list)**

The `(list)` should be a comma-separated list of fields that have values in the form `[⟨tag⟩]⟨id-list⟩` where `⟨id-list⟩` is a comma-separated list of entry labels. The value is required for this key but may be empty, which indicates an empty set of fields (that is, the setting is switched off).

This setting makes those fields act like the `see` field by identifying the listed entries as dependencies, but the information isn’t added to the cross-reference part of the location list. This action is performed after `labelify-list`, if that’s also set.

For example, suppose the file `entries-en.bib` contains:

```latex
@index{cat,
    translations-pt={gato,gatinho},
    seealso={kitten}
}

@index{kitten,
    translations-pt={gato,gatinho}
}

@index{staple}
@index{rivet}
@index{mat}
@index{carpet}
@index{rug}
@index{tapestry}
@index{doormat}
@index{matting}
```
and suppose the file entries-pt.bib contains:

@index{gato,
    prefix={o},
    translations-en={cat, staple, rivet},
    seealso={gatinho}
}

@index{gatinho,
    translations-en={kitten}
}

@index{tapete,
    translations-en={carpet, rug, mat, tapestry}
}

@index{esteira,
    prefix={a},
    translations-en={mat, track, matting, furrow}
}

@index{capacho,
    prefix={o},
    translations-en={doormat, matting, mat, coconut-matting}
}

The aim here is to have a document containing an English-to-Portuguese and a Portuguese-to-English dictionary. The custom translations-pt and translations-pt fields contain comma-separated lists of possible translations. In this case I don’t want to use the see field (and, in fact, can’t for the entries that have the seealso field set), but I can identify the values of those fields as dependent entries to ensure that they are selected even if they’re not referenced in the document.

For convenience I’ve aliased the custom fields to user1:

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[british, brazilian]{babel}

167
5.7 Field and Label Options

\usepackage[colorlinks]{hyperref}
\usepackage[record,nomain,nostyles,stylemods={bookindex},style={bookindex}]{glossaries-extra}
\usepackage{glossaries-prefix}
\newglossary*{en}{English Terms}
\newglossary*{pt}{Portuguese Terms}

\GlsXtrLoadResources[
  type={en},
  src={entries-en},
  sort={en-GB},
  category={en},
  field-aliases={translations-pt=user1},
  dependency-fields={user1},
  sort-label-list={user1:pt-BR:glsentryname}
]
\GlsXtrLoadResources[
  type={pt},
  src={entries-pt},
  sort={pt-BR},
  category={pt},
  field-aliases={translations-en=user1},
  dependency-fields={user1},
  sort-label-list={user1:en-GB:glsentryname}
]
\apptoglossarypreamble[en]{\selectlanguage{british}}
\apptoglossarypreamble[pt]{\selectlanguage{brazilian}}

\begin{document}
\selectlanguage{british}
The \texttt{\gls{cat}} sat on the \texttt{\gls{mat}}.
\selectlanguage{brazilian}O \texttt{\gls{gato}} sentou-se no \texttt{\gls{tapete}}.

\renewcommand*{\glsxtrbookindexname}[1]{%
\glsxtrifhasfield{prefix}{#1}{\xmakefirstuc\glscurrentfieldvalue\texttt{\_}}{}}%
\glossentryname{#1}%

168
5.7 Field and Label Options

Assignments

group=(label)

The group option will set the group field to ⟨label⟩ unless ⟨label⟩ is auto. If group={auto} then if the --group switch is used the value of the group field is set automatically during the sorting (see also group-formation, group-level and section 1.2). If the --no-group setting is on then group={auto} does nothing.

The corresponding group title can be set with \glsxtrsetgrouptitle in the document if the title is different from the label. The default behaviour is group={auto}.

For example:

\GlsXtrLoadResources[sort={integer},group={Constants},
src={entries-constants}% data in entries-constants.bib
]
\GlsXtrLoadResources[sort={letter-case},group={Variables},
src={entries-variables}% data in entries-variables.bib
]

In this case, if the type field hasn’t been set in the .bib files, these entries will be added to the same glossary, but will be grouped according to each instance of \GlsXtrLoadResources, with the provided group label.

category=(value)

The selected entries may all have their category field changed before writing their definitions to the .glstex file. The ⟨value⟩ may be:

- false: switch off this setting (default);
- same as entry: set the category to the .bib entry type used to define it (lower case and without the initial @) after any aliasing, if applicable;
- same as original entry: (new to v1.4) set the category to the original entry type (lower case and without the initial @) before it was aliased (behaves like same as entry if the entry type wasn’t aliased);
- same as base: (new to v1.1) set the category to the base name of the .bib file (without the extension) that provided the entry definition;
5.7 Field and Label Options

- `same as type`: set the `category` to the same value as the `type` field (if that field has been provided either in the `.bib` file or through the `type` option);

- `{label}`: the `category` is set to `{label}` (which mustn’t contain any special characters).

This will override any `category` fields supplied in the `.bib` file.

When used with `entry-type-aliases`, the option `category={same as entry}` refers to the target entry type whereas `category={same as original entry}` refers to the original entry type given in the `.bib` file. In both cases, the value is converted to lower case to ensure consistency. An alternative is to use `save-original-entrytype={category}`. When combined with `save-original-entrytype-action={changed}` it’s then possible to only set the `category` to the original entry type for aliased entries and leave it unmodified for unaliased entries.

For example, if the `.bib` file contains:

```latex
@entry{bird,
  name={bird},
  description = {feathered animal}
}

@index{duck}

@index{goose,plural="geese"}

@dualentry{dog,
  name={dog},
  description={chien}
}
```

then if the document contains:

```latex
\GlsXtrLoadResources[category={same as entry},src={entries}]
```

this will set the `category` of the bird term to `entry` (since it was defined with `@entry`), the `category` of the duck and goose terms to `index` (since they were defined with `@index`), and the `category` of the dog term to `dualentry` (since it was defined with `@dualentry`). Note that the dual entry `dual.dog` doesn’t have the category set, since that’s governed by `dual-category` instead.

If, instead, the document contains:

```latex
\GlsXtrLoadResources[category={animals},src={entries}]
```

then the `category` of all the primary selected entries will be set to `animals`. Again the dual entry `dual.dog` doesn’t have the `category` set.

Note that the categories may be overridden by the commands that are used to actually define the entries (such as `\bibglsnewindex`).

For example, if the document contains:

170
5.7 Field and Label Options

\newcommand{\bibglsnewdualentry}[4]{%
  \longnewglossaryentry*{#1}{name={#3},#2,category={dual}}{#4}%
}

\GlsXtrLoadResources[category={animals},src={entries}]
then both the dog and dual.dog entries will have their category field set to dual since the new definition of \bibglsnewdualentry has overridden the category={animals} option.

\texttt{type={⟨value⟩}}

The \texttt{⟨value⟩} may be one of:

- \texttt{false}: switches off this setting (default);
- \texttt{same as entry}: set the \texttt{type} field to the entry type (lower case and without the initial @);
- \texttt{same as original entry}: set the \texttt{type} to the original entry type (lower case and without the initial @) before it was aliased (behaves like same as entry if the entry type wasn’t aliased);
- \texttt{same as base}: set the \texttt{type} field to the base name of the corresponding .bib file (without the extension);
- \texttt{same as category}: set the \texttt{type} field to the same value as the category field (type unchanged if category not set);
- \texttt{same as parent}: sets the \texttt{type} to the same as the entry’s parent (new to v1.9). If the entry doesn’t have a parent or if the parent doesn’t have the \texttt{type} field set, then no change is made. Entries should always have the same type as their parent, but it’s possible for spawned entries to pick up the \texttt{type} field from their progenitor entry (if it was explicitly set in the .bib file), which may be inappropriate.
- \texttt{⟨label⟩}: sets the \texttt{type} field to the glossary identified by \texttt{⟨label⟩}.

When used with \texttt{entry-type-aliases}, the option \texttt{type={same as entry}} refers to the target entry type and \texttt{type={same as original entry}} refers to the original entry type given in the .bib file. An alternative is to use \texttt{save-original-entrytype={type}}. When combined with \texttt{save-original-entrytype-action={changed}} it’s then possible to only set the \texttt{type} to the original entry type for aliased entries and leave it unmodified for unaliased entries.

It’s not possible to have both \texttt{category={same as type}} and \texttt{type={same as category}}.

Note that this setting only changes the \texttt{type} field for primary entries. Use \texttt{dual-type} for dual entries.
For example:
5.7 Field and Label Options

\usepackage[record,symbols]{glossaries-extra}

\GlsXtrLoadResources[src={entries-symbols},type={symbols}]

Make sure that the glossary type has already been defined (see section 1.3). In the above, the symbols option defines the symbols glossary. If you want to use a custom glossary, you need to provide it. For example:

\usepackage[record,nomain]{glossaries-extra}
\newglossary*[dictionary]{Dictionary}
\GlsXtrLoadResources[src={entries-symbols},type={dictionary}]

(The nomain option was added to suppress the creation of the default main glossary.)

trigger-type=⟨type⟩

The record counting commands, such as \rgls, use the special format \glstriggerrecord-format, which bib2gls also treats as an ignored record. This means the entry will still be identified as having a record for selection purposes, which is necessary for the entry to be defined for use in the document, but in order to prevent it from appearing in the glossary you need to transfer the entry with trigger-type=⟨⟨⟨type⟩⟩⟩. This will override the type, dual-type, tertiary-type and the type specification in secondary.

The provided value ⟨⟨⟨type⟩⟩⟩ must be a glossary label (not one of the keywords allowed by type) or false to switch off this setting. You can define the glossary before loading the resource, but it’s not required as bib2gls will write \provideignoreglossary*{⟨⟨⟨type⟩⟩⟩} to the .glstex file even if --no-provide-glossaries is set (see section 1.3).

progenitor-type=⟨type⟩

This sets the default type field for the main term defined by @progenitor-like entries. The ⟨value⟩ is as for type. This doesn’t change the type for the spawned progeny.

progeny-type=⟨type⟩

This sets the default type field for the progeny term spawned by @progenitor-like entries. The ⟨value⟩ is as for type. This doesn’t change the type for the main progenitor. Remember that with the default adopted-parent-field={parent} setting, the given type should match the type of the parent entry.

adopted-parent-field=⟨type⟩

This identifies the target field to be set to the corresponding value of the adoptparents list by the progeny entries spawned by the @progenitor type of entry. The default is parent.
ignore-fields=⟨list⟩

The ignore-fields key indicates that you want bib2gls to skip the fields listed in the supplied comma-separated ⟨list⟩ of field labels. Remember that unrecognised fields will always be skipped. However, an unrecognised field can still be referenced with some options (such as replicate-fields) whereas any field excluded with ignore-fields will be discarded and can’t be referenced.

This setting is always implemented after field-aliases (see section 1.4). If a field has been aliased then the original field name is no longer present and so ignoring it will have no effect.

For example, suppose my .bib file contains:

```latex
@abbreviation{html,
  short = "html",
  long  = {hypertext markup language},
  description={a markup language for creating web pages},
  seealso={xml}
}
```

but I want to use the short–long style and I don’t want the cross-referenced term, then I can use ignore-fields={seealso,description}.

Note that ignore-fields={parent} removes the parent before determining the dependency lists. This means that selection={recorded and deps} and selection={recorded and ancestors} won’t pick up the label in the parent field.

If you want to maintain the dependency and ancestor relationship but omit the parent field when writing the entries to the .glstex file, you need to use flatten instead.

field-aliases=⟨key=value list⟩

You can instruct bib2gls to treat one field as though it was another using this option. The value should be a comma-separated list of ⟨field1⟩=⟨field2⟩ pairs, where ⟨field1⟩ and ⟨field2⟩ are field names. Identical mappings and trails aren’t permitted. (That is, ⟨field1⟩ and ⟨field2⟩ can’t be the same nor can you have both ⟨field1⟩=⟨field2⟩ and ⟨field2⟩=⟨field3⟩.) If you want to swap fields you need to use one of the dual entry types instead. Field aliases are performed before ignore-fields, so if ⟨field1⟩ is listed in ignore-fields it won’t be ignored (unless ⟨field2⟩ is in ignore-fields).

For example, suppose people.bib contains:

```latex
@entry{alexander,
  name={Alexander III of Macedon},
  description={Ancient Greek king of Macedon},
  born={20 July 356 BC},
  died={10 June 323 BC},
  othername={Alexander the Great}
}
```

173
5.7 Field and Label Options

This contains three non-standard fields: born, died and othername. I could define these fields using \glsaddkey, but another possibility is to map these onto the user keys user1, user2 and user3, which saves the overhead of providing new keys:

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

\GlsXtrLoadResources[
  src={people},% data in people.bib
  field-aliases={born=user1,died=user2,othername=user3}
]

Note the difference in syntax between replicate-fields and assign-fields. Both have a key=value list as the option argument, but the \{key\}=\{value\} syntax is different. In the case of replicate-fields, the left hand side (\{key\}) is the source field. The right hand side (\{value\}) is a comma-separated list of destination fields. The value of the source field will be copied into each of the destination fields. In the case of assign-fields, the left hand side (\{key\}) is the destination field and the right hand side value is an assignment expression with an optional conditional.

The value of one field can be copied to other fields using this option where each \{key\}=\{value\} pair is in the form \{field1\}={\{field2\}, \{field3\},…} where all values are field names. The value is required for this key but may be empty, which indicates that the setting is switched off.

This option copies the contents of \{field1\} to \{field2\}, \{field3\}, … (but only if the target field isn’t already set with replicate-override={false}). This action is performed after ignore-fields (see section 1.4). If the source field is missing, the replicate-missing-field-action setting determines the action.

If the target field doesn’t have an associated key recognised by \newglossaryentry then the value will be saved using \GlsXtrSetField. Special internal fields aren’t permitted as either source or target fields.

For example, suppose people.bib contains:

\@entry{alexander,
  name={Alexander III of Macedon (Alexander the Great)},
  text={Alexander},
  description={Ancient Greek king of Macedon}
}

Since the first field hasn’t been supplied, it will default to the value of the text field, but perhaps for one of my documents I’d like the first field to be the same as the name field. Rather than editing the .bib file, I can just do:

\GlsXtrLoadResources[
  src={people},% data in people.bib
  replicate-fields={name=first}
]
This copies the contents of the name field into the first field. If you have more than one field in the list take care to brace the lists to avoid confusion. For example, if for some reason I want to copy the value of the name field to both first and firstplural and copy the value of the text field to the plural field, then this requires braces for the inner list:

\GlsXtrLoadResources[
  src={people},% data in people.bib
  replicate-fields={name={first,firstplural},text=plural}
]

If my people.bib file instead contained:

@entry{alexander,
  name={Alexander III of Macedon (Alexander the Great)},
  first={Alexander the Great},
  text={Alexander},
  description={Ancient Greek king of Macedon}
}

then:

\GlsXtrLoadResources[
  src={people},% data in people.bib
  replicate-fields={name=first}
]

won’t alter the first field since replicate-fields doesn’t override existing values by default. You can use replicate-override to change this. Alternatively, since replicate-fields is always performed after ignore-fields it’s possible to ignore the first field which means that the name value can then be copied into it:

\GlsXtrLoadResources[
  src={people},% data in people.bib
  ignore-fields={first},
  replicate-fields={name=first}
]

Note that the ordering within the resource options doesn’t make a difference. The same result occurs with:

\GlsXtrLoadResources[
  src={people},% data in people.bib
  replicate-fields={name=first},
  ignore-fields={first}
]
5.7 Field and Label Options

replicate-override={⟨boolean⟩}

This is a boolean option. The default setting is replicate-override={false}. If true, replicate-fields will override the existing value if the target field is already set.

replicate-missing-field-action={⟨value⟩}

This option indicates what to do if a source field identified in replicate-fields is missing. The value may be one of:

- `skip`: skip the replication of the missing field (default);
- `fallback`: use the fallback for the missing field (see section 5.8), if one is available (otherwise skip);
- `empty`: make the target field empty.

assign-fields=⟨key=value list⟩

Note the difference in syntax between replicate-fields and assign-fields, as highlighted in the replicate-fields section, above. The assign-fields option is implemented after the replicate-fields option (see section 1.4).

The assign-fields option is a more complicated way of setting a field than replicate-fields. Each 〈key=value list〉 element of the key=value list argument has the syntax:

〈dest-field〉 = [〈override〉] ⟨element-list⟩ [〈condition〉]

If the destination field (〈dest-field〉) is already set, it will only be overwritten if assign-override={true} or if [override] is “o”. The 〈dest-field〉 is simply the name of the field for the entry under consideration and doesn’t use the more complex 〈field-ref〉 syntax used in 〈element-list〉, which is described in section 5.1.

The [〈override〉] following the equal sign is optional and may be used to counteract the assign-override setting for the given assignment. The 〈override〉 value may be either “o” (override) or “n” (no override). If not present, the assign-override setting will be used.

The 〈element-list〉 is a string concatenation, as described in section 5.1. If any individual element in the list evaluates to null, the entire string is deemed to be null, in which case the assignment won’t be made.

The [〈condition〉] part is optional. If present, the assignment is only made if the condition evaluates to true. The condition should be placed in square brackets after the 〈element-list〉 part. This is a complex conditional, as described in section 5.2.

Note that, unlike most 〈key=value〉 options, the value part (〈element-list〉 [〈condition〉]) should not be grouped. The assign-fields option is parsed in a different way to the other key=value list options. However, it’s best to group the entire 〈value〉 argument of assign-fields. For example:
assign-fields=
  name = text + ", " + symbol
}

Don’t do name = {text + ", " + symbol}.

Remember that field values may be altered before or after assign-fields by other resource options (see section 1.4). The assignment will use the value current at the time it is referenced during the processing of assign-fields. If you need to reference the destination field in the assignment, make sure that the override setting is on if the field needs to be updated.

For example, suppose I have defined the custom fields surname and forename, and I have the following in my .bib file:

```latex
@index{Smith}
@index{Jane-Smith,
  forename={Jane},
  parent={Smith}
}
```

Suppose that what I actually want is:

```latex
@index{Smith}
@index{Jane-Smith,
  forename={Jane},
  surname={Smith},
  parent={Smith},
  name={Jane},
  text={Jane Smith}
}
```

This is quite repetitive to type out for every person you need to index. The replicate-fields option can reduce some of this. For example:

```latex
replicate-fields=
  forename={name},
  surname={parent}
}
```

This doesn’t deal with the text field and also has a problem if the parent field (which should contain a label) doesn’t match the surname. For example, I might also have:

```latex
@index{de-la-Fontaine,
  name={de la Fontaine}
}
```
5.7 Field and Label Options

In this case the custom `surname` field needs to match the parent’s `name` field, not the parent’s label.

The desired result can instead be obtained with:

```latex
assign-fields={
   surname = parent -> name,
   name = self -> forename,
   text = self -> forename
      + " " + self -> surname
}
```

The `self ->` part is optional so this can be written more compactly as:

```latex
assign-fields={
   surname = parent -> name,
   name = forename,
   text = forename + " " + surname
}
```

The last assignment in the above can also be written as:

```latex
text = forename + { } + surname
```

Suppose, for some reason, I want the first use to show the surname in bold. This means I need to add `\textbf{` before the value of the surname field and the closing `}` needs to go after. This can be achieved with:

```latex
first = forename + " \textbf{" + surname "}"\textbf{"
```

Note that because there are unbalanced braces in the string fragments, it’s necessary to use quote delimiters. Since `\textbf` is robust, there’s no need to protect it from expansion.

Suppose, instead, I want the surname in upper case on first use. I could simply replace `\textbf` with `\MakeUppercase`, but I can get bib2gls to do the case-conversion instead:

```latex
first = forename + " " + \UC{ surname }
```

This assumes that `\GlsXtrResourceInitEscSequences` has been added to the definition of `\glsxtrresourceinit`, as described in section 1.5. Otherwise you would need to protect `\UC`.

In the above example, the `surname` field is obtained from the value of the parent’s `name` according to the assignment:

```latex
surname = parent -> name,
```
In the case of the Smith entry, the name field hasn’t been set.

If a referenced field hasn’t been set then the behaviour depends on the assign-missing-field-action setting. The default behaviour is to use the fallback, if provided (see section 5.8). In the case of @index, the fallback for the name field is the entry’s label. This means that the Jane-Smith entry will have the surname field set to “Smith”.

If the fallback isn’t set or if there is no fallback then the assignment instruction will be skipped. Similarly, if an ancestor is referenced but doesn’t exist then assignment will again be skipped.

The ancestor entries must be defined first to ensure that they have been processed and have had their fields set before they can be referenced in an assignment.

Since I haven’t imposed any conditions on the assignments, the instructions will be attempted on all entries. This includes the parent entries.

The first instruction attempts to set the surname field to the parent’s name. Neither the Smith nor the de-la-Fontaine entries have a parent entry, so this instruction is skipped for both of them.

The second instruction attempts to set the name field to the entry’s forename field. The de-la-Fontaine entry already has the name field set so the instruction is automatically skipped (because of the default assign-override={false}). The Smith entry doesn’t have the name field set so the assignment will be attempted but will fail because the forename field isn’t set and doesn’t have a fallback.

The text (and first) instruction similarly references the forename field, which isn’t set, so the instruction is skipped. The instruction also contains a reference to the surname field, but that part of the assignment isn’t reached as the attempt stops at the first unset reference.

This means that neither the Smith nor the de-la-Fontaine entries will be affected by the above assign-fields setting.

Each instruction will be attempted, in turn, unless the assign-override setting prevents it. This means it’s possible to have multiple assignments for a particular field. The first one that succeeds will be the one that has effect (with assign-override={false}). For example:

```plaintext
assign-fields=
  surname = parent -> name,
  surname = name,
  name = forename,
  text = forename + " " + surname
}
```

This has two instructions for the surname. The first one will be applied to entries that have a parent and the second one will be applied to the other entries (that don’t already have the surname set).

Suppose I also have some monarchs, who are more typically only referred to by their forename (with no surname), possibly prefixed by their rank and suffixed by their ordinal.
Let’s further suppose that in my document I have also defined the custom fields `rank` and `ordinal`, and in my `.bib` file I have:

```bibtex
@indexplural{king}
@indexplural{queen}
@indexplural{empress,plural={empresses}}
@index{King-Stephen, 
    parent={king}, 
    forename={Stephen}}
@index{Empress-Matilda, 
    parent={empress}, 
    forename={Matilda}, 
}
@index{Elizabeth-I, 
    parent={queen}, 
    forename={Elizabeth}, 
    ordinal=I 
}

The earlier assignment rules won’t be appropriate for these cases, as they would result in the text fields: “Stephen kings”, “Matilda empresses” and “Elizabeth queens”.

In this case, the assignment rules need to depend on the type of entry. I could test if the parent label is “king” or “empress” or “queen”, but a more reliable approach is to have custom entry types which can be aliased:

```bibtex
@index{Smith}
@person{Jane-Smith, 
    forename={Jane}, 
    parent={Smith}}
@index{de-la-Fontaine, 
    name={de la Fontaine}}
@person{Margaret-de-la-Fontaine, 
    forename={Margaret}, 
    parent={de-la-Fontaine}}
@indexplural{king}
@indexplural{queen}
@indexplural{empress,plural={empresses}}
@monarch{King-Stephen, 

The resource options are now:

```plaintext
entry-type-aliases={person=index,monarch=index},
assign-fields={
    name = forename + `''-'' ordinal,
    name = forename,
    surname = parent -> name
    [ entrytype -> original = "person" ],
    text = forename + " " + surname
    [ entrytype -> original = "person" ],
    first = \FIRSTUC { parent `''follow'' text } + " " + name
    [ entrytype -> original = "monarch" ],
    text = name
    [ entrytype -> original = "monarch" ]
}
```

Additional fields can be added to accommodate nicknames or other forms of address, which will add to the complexity of the assignment specification.

```plaintext
assign-override={⟨boolean⟩}
```

This is a boolean option. The default setting is `assign-override={false}`. If `true`, `assign-fields` will override the existing value if the target field is already set.

```plaintext
assign-missing-field-action={⟨value⟩}
```

This option indicates what to do if a source field identified in `assign-fields` is missing. The value may be one of:

- `skip`: skip the assignment;
- `fallback`: use the fallback for the missing field (see section 5.8), if one is available, otherwise skip the assignment (default);
5.7 Field and Label Options

- empty: treat the missing value as empty.

Return null will result in the assignment instruction being omitted.

counter={value}

The counter option assigns the default counter to use for the selected entries. (This can be overridden with the counter key when using commands like \gls.) The value must be the name of a counter. Since bib2gls doesn’t know which counters are defined within the document, there’s no check to determine if the value is valid (except for ensuring that ⟨value⟩ is non-empty). A value of false will switch off this setting (the default).

Note that this will require an extra \LaTeX and bib2gls call since the counter can’t be used for the indexing until the entry has been defined.

copy-action-group-field={value}

This option may only be used when invoking bib2gls with the --group (or -g) switch. If an action other than the default action={define} is set, this option can be used to identify a field in which to save the letter group information where ⟨value⟩ is the name of the field. This just uses \GlsXtrSetField. You will need to redefine \glsxtrgroupfield to ⟨value⟩ before displaying the glossary. For example, if copy-action-group-field={dupgroup}, action={copy} and type={copies} are set in the resource options and copies identifies a custom glossary:

\printunsrtglossary*[type={copies},style={indexgroup}]
{\renewcommand{\glsxtrgroupfield}{dupgroup}}

This option is ignored when used with action={define}. This option is not used by secondary which will always save the group information in the secondarygroup field. When used with action={define or copy}, entries that are defined will have both group and the field given by copy-action-group-field set.

Note that you may do copy-action-group-field={group} which will override the group field from the original definition. This may be useful if you don’t use grouping in the primary glossary. That is, you use nogroupskip and a non-group style. For example:

\printunsrtglossary[nogroupskip,style={index}]
\printunsrtglossary[type={copies},style={indexgroup}]

copy-alias-to-see={boolean}

If set, the value of the alias field is copied to the see field. The default setting is copy-alias-to-see={false}.
5.7 Field and Label Options

save-from-see=⟨value⟩

This option allows you to save a comma-separated list of entry labels in a designated internal field of the target entry identified by their see field. If the ⟨value⟩ is omitted, save-from-see={from-see} is assumed. The value may be the keyword false, which switches off this setting, otherwise the value should be the desired name of the internal field. The default setting is save-from-see={false}.

For example, if the .bib file contains:

@index{gourd}
@index{cucumber,see={gourd}}
@index{pumpkin,see={gourd}}

then the resource option save-from-see={from-see} will create an internal field called from-see for the gourd entry that contains the comma-separated list cucumber,pumpkin.

Note that the given internal field isn’t actually assigned within bib2gls, so it can’t be accessed via any resource options. Each item in this list is added using \glsxtrapptocsv-field after the source entry (that is, the entry containing the see field) is defined in the .glstex file. This means that the list will be in the same order as the entries. You can then pass the field value to \glsseelist. For example:

\glsdefpostdesc{%
\glsxtrifhasfield{from-see}{\glscurrententrylabel}{\glscurrentfieldvalue}
, related: \glsseelist{\glscurrentfieldvalue}}

This option has no effect with the “no dependency” selection criteria (such as selection ={recorded no deps}).

save-from-seealso=⟨value⟩

As save-from-see but for theseealso field. If the value is omitted, save-from-seealso ={from-seealso} is assumed.

save-from-alias=⟨value⟩

As save-from-see but for the alias field. If the value is omitted, save-from-alias= {from-alias} is assumed.

save-crossref-tail=⟨value⟩

If you have a cross-reference trail where one entry references another entry using see, seealso or alias and the referenced entry also references another, and so on, then you can save the tail end of the trail with this option. Note that the trail only follows single-label lists (in see or seealso). The trail is terminated if an entry doesn’t have one of those three fields set or if it cross-references multiple entries or if the trail loops back on itself.

If you have a loop, the tail for some entries may end prematurely since the algorithm to obtain the tail saves the tail for each sub-trail to avoid recalculating it. It’s best to avoid this
setting if you have cross-reference loops. (Aside from two-way cross-references, it’s best to
avoid loops in general.)

The tail label is stored in the field identified by the \langle value\rangle of this option. If the value
is omitted, save-crossref-tail={crossref-tail} is assumed. The field won’t be set if
there’s no tail. The tails are calculated when writing the entry definitions to the .glstex file
so the value can’t be referenced or otherwise accessed by bib2gls.

Example:

@index{sample1,see={sample2}}
@index{sample2,see={sample3}}
@index{sample3,see={sample4}}
@index{sample4}

The tail for sample1 is sample4. As a by-product of the recursion used in calculating the tail
for sample1, the tail for each element in the trail (sample2 and sample3) is also calculated.
The tail is the same for each entry in the trail. The final entry sample4 doesn’t have a tail.

If sample4 is modified to cross-reference sample1:

@index{sample4,see={sample1}}

then when the tail for sample4 is calculated the tail for its cross-reference (sample1) is
consulted. This has already been set to sample4. An entry can’t have itself as a tail so
the tail for sample4 is set to sample3. All the other entries still have sample4 as their tail
because their tail was determined while traversing the trail for sample1, which had to stop
when it wrapped round to its starting point.

save-original-entrytype=(value)

The \langle value\rangle may be the keywords false or true or the name of a field in which to store
the original entry type (as given in the .bib file but without the leading @ and converted
to lower case). The setting is save-original-entrytype={false}. If \langle value\rangle is omitted
or the keyword true, then save-original-entrytype={originalentrytype} If \langle value\rangle
has an associated key in newglossaryentry (for example, one provided with \glsadd-
storagekey) it will be set after the field aliases, otherwise (for example, originalentry-
type) it will simply be added to the .glstex file using \GlsXtrSetField after the entry
definition (which means the field can’t be referenced in other resource options). This setting
is governed by save-original-entrytype-action.

save-original-entrytype-action=(value)

This option determines whether or not save-original-entrytype should save the original
entry type. No action is performed when save-original-entrytype={false} otherwise
the action is determined by \langle value\rangle which may be one of the following keywords:

- always: always save the original entry type (default);
- no override: don’t override a field that’s already been set;
5.7 Field and Label Options

- **changed override** or **changed or diff**: only save the original entry type if it’s different from the final entry type;

- **changed no override**: only save the original entry type if it’s different from the final entry type and the specified field hasn’t been set.

The “no override” options make no difference if the given field is unknown (such as `original-entrytype`). For known fields, bear in mind that the field will be set after field aliasing but before other options, such as `ignore-fields`.

The “changed” options ignore case. For example, if the `.bib` file defined an entry with `@INDEX` then both the original and final entry type will be `index`.

**Field Adjustments**

`post-description-dot={value}`

The `postdot` package option (or `nopostdot={false}`) can be used to append a full stop (.) to the end of all the descriptions. This can be awkward if some of the descriptions end with punctuation characters. This resource option can be used instead. The `{value}` may be one of:

- **none**: don’t append a full stop (default);

- **all**: append a full stop to all `description` fields in this resource set;

- **check**: selectively append a full stop (see below).

Note that if you have dual entries and you use this option to append a full stop, then it will be copied over to the mapped field. This is different to the `postdot` option which doesn’t add the dot to the field but incorporates it in the post-description hook. This means that a dot inserted with `post-description-dot` will come before the post-description hook whereas with `postdot` the punctuation comes after any category-specific hook.

The `post-description-dot={check}` setting determines whether to append the dot as follows:

- If the `description` field ends with \npostdesc or \glsxtrnopostpunc, then a dot isn’t appended.

- If the `description` field doesn’t end with a regular (ungrouped letter or other) character, then a dot is appended. (For example, if the description ends with a control sequence or an end group token.)

- If the `description` field ends with a character that belongs to the Unicode category “Punctuation, Close” or “Punctuation, Final quote” then the token preceding that character is checked.

- If the `description` field doesn’t end with a character that belongs to the Unicode category “Punctuation, Other” then the dot is added.
Note that the interpreter isn’t used during the check. If the `description` ends with a command then a dot will be appended (unless it’s \glsxtrnopostpunc or \nopostdesc) even if that command expands in such a way that it ends with a terminating punctuation character. This option only applies to the `description` field.

`strip-trailing-nopost=(boolean)`

This option is always performed before `post-description-dot`. The default setting is `strip-trailing-nopost={false}`. If `true` any trailing ungrouped \nopostdesc or \glsxtrnopostpunc found in the `description` field will be removed. Note that the command (possibly followed by ignored space) must be at the very end of the description for it to be removed. A description should not contain both commands. This option only applies to the `description` field.

For example, \nopostdesc will be stripped from:

```
description={sample\nopostdesc}
```

since it’s at the end. It will also be stripped from:

```
description={sample\nopostdesc }
```

since the trailing space is ignored as it follows a control word. It won’t be stripped from:

```
description={sample\nopostdesc{}}
```

because the final space is now significant, but even without the space it still won’t be stripped as the field ends with an empty group not with \nopostdesc. Similarly it won’t be stripped from:

```
description={sample\nopostdesc\relax}
```

because again it’s not at the end.

`check-end-punctuation=(list)`

This options checks the end of all the fields given in `(list)` for end of sentence punctuation. This is determined as follows, for each `(field)` in the comma-separated `(list)`:

- if the last character is of type “Punctuation, Close” or “Punctuation, Final quote”, check the character that comes before it;
- if the character is of type “Punctuation, Other”, then check if it’s listed in the entry given by `sentence.terminators` in `bib2gls`’s language resource file.

If a sentence terminator is found, an internal field is created called `(field)endpunc` that contains the punctuation character. Fields whose values must be labels (such as `parent`, `category` and `type`) aren’t checked, even if they’re included in `(list)`.

The default `sentence.terminators` is defined in `bib2gls-en.xml` as:
Any character that isn’t of type “Punctuation, Other” won’t match.
For example, the sample books.bib file contains:

```biblatex
@entry{whydidnttheyaskevans,
    name={Why Didn't They Ask Evans?},
    description={novel by Agatha Christie},
    identifier={book},
    author={\sortmediacreator{Agatha}{Christie}},
    year={1934}
}
```

With `check-end-punctuation={name}`, this entry will be assigned an internal field called `nameendpunc` set to `?` as that’s included in `sentence.terminators` and is found at the end of the `name` field:

```latex
\GlsXtrSetField{whydidnttheyaskevans}{nameendpunc}{?}
```

(Note that `check-end-punctuation={first,text}` won’t match as there’s no `first` or `text` field supplied.)

If you have a field that ends with an abbreviation followed by a full stop, this will be considered an end of sentence terminator, but the main purpose of this option is to provide a way to deal with cases like:

Agatha Christie wrote \gls{whydidnttheyaskevans}.

where the end of sentence punctuation following \gls needs to be discarded. This is needed regardless of whether or not the link text ends with an abbreviation or is a complete sentence.

It’s then possible to hook into the post-link hook “discard period” check. By default this just checks the category attributes that govern whether or not to discard a following period, but (with glossaries-extra v1.23+) it’s possible to provide an additional check by redefining:

```latex
\glsxtrifcustomdiscardperiod{⟨true⟩}{⟨false⟩}
```

This should expand to ⟨true⟩ if the check should be performed otherwise it should expand to ⟨false⟩. You can reference the label using \glslabel. For example:

```latex
\renewcommand*{\glsxtrifcustomdiscardperiod}[2]{% \GlsXtrIfFieldUndef{nameendpunc}{\glslabel}{#2}{#1}% }
```

This uses \GlsXtrIfFieldUndef rather than \glsxtrifhasfield* since there’s no need to access the field’s value. (The unstarred form \glsxtrifhasfield can’t be used as it introduces implicit scoping, which would interfere with the punctuation lookahead.)

The other difference between \GlsXtrIfFieldUndef and the other \...hasfield tests is the case where the field is set to an empty value. In this case the field is defined (so \GlsXtrIfFieldUndef does the ⟨false⟩ argument) but it’s considered unset (so commands like \ifglshasfield do the ⟨false⟩ argument).
5.7 Field and Label Options

sort-label-list=⟨list⟩

This option takes a list as the value with each element in the list in the form:

⟨field-list⟩: ⟨sort⟩: ⟨csname⟩

or:

⟨field-list⟩: ⟨sort⟩

where:

- ⟨field-list⟩ is a comma-separated list of valid fields;
- ⟨sort⟩ is a valid sort method as per the sort option, but not including none or unsrt;
- ⟨csname⟩ is the name (without a leading backslash) of a command that takes a label as its sole mandatory argument that’s recognised by bib2gls’ interpreter (such as those listed in table 2.1).

The final : ⟨csname⟩ part may be omitted if no command need be applied. (That is, sort by label.) The value is required for this key but may be empty, which indicates the setting is switched off.

The sorting options are as those for the main list. For example, for entries in the primary list the break point is obtained from the break-at setting and for entries in the dual list the break point is obtained from dual-break-at. (Remember that if dual-sort={combine} then there is only one list that contains both the primary and dual entries, which is governed by the primary options only.)

If the ⟨field-list⟩ has more than one element take care to use braces {} to avoid confusion for the list-parser. For example:

\GlsXtrLoadResources[
    sort-label-list=\{\langle see,seealso\rangle:en:glsentryname\}
]

Note that strange results may occur if this setting is used on any fields that don’t simply contain a list of entry labels or if any of the referenced entries are processed in different resource sets (see section 1.4).

After the main sorting of each set of selected entries is performed (as per sort or dual-sort), if this option is set, then for each {⟨field-list⟩}: ⟨sort⟩: ⟨csname⟩ the following steps are performed:

1. For each entry ⟨id⟩:

   a) For each ⟨field⟩ in ⟨field-list⟩, if the field is set for entry ⟨id⟩ then:

      i. The field value must be in the form ⟨⟨tag⟩⟩ ⟨label-list⟩ where ⟨⟨tag⟩⟩ is optional and ⟨label-list⟩ is a comma-separated list of entry labels ⟨label₁⟩, ..., ⟨labelₙ⟩;
ii. A new list is constructed where the \textit{i}th element is: \{\texttt{\textless csname\textgreater \texttt{\{label\}}\}\} unless \texttt{\textless csname\textgreater} hasn’t been set, in which case the \textit{i}th element is just \{\texttt{\textless label\textgreater}\} (the optional \texttt{\textless tag\textgreater} part is omitted);

iii. This new list is sorted according to the interpreter’s definition of the command given by \texttt{\textless csname\textgreater} (if provided) and the designated \texttt{\textless sort\textgreater} method;

iv. The field value is reconstructed with the labels in the corresponding order (prefixed with \texttt{\textless tag\textgreater} if it was present in the original).

Note that there is no hierarchical structure in the sorting of the field list even if any of the referenced entries has a parent.

For example, suppose the file \texttt{entries.bib} contains:

```bibtex
@index{bird}
@index{waterfowl, parent={bird} }
@index{duck,  
        parent={waterfowl},  
        seealso={swan,duckling,parrot,goose} }
@index{swan,  
        parent={waterfowl},  
        seealso={goose,duck} }
@index{goose,  
        parent={waterfowl},  
        seealso={duck} }
@index{parrot, parent={bird} }
@index{duckling,  
        see={\texttt{[related terms]}fluffy,velociraptor,duck,tardigrade} }
@index{fluffy}
@index{tardigrade, name={water bear} }
@index{velociraptor}
```

And suppose the document contains:
5.7 Field and Label Options

\documentclass{article}
\usepackage[record,style={tree}]{glossaries-extra}
\GlsXtrLoadResources[
  src={entries},
  sort={en},
  sort-label-list={{seealso,see}:en:glsentryname}
]
\begin{document}
\Gls{parrot}, \gls{tardigrade}, \gls{swan}, \gls{duck}, \gls{goose}, \gls{fluffy} \gls{duckling}, \gls{velociraptor}.
\printunsrtglossaries
\end{document}

Then this reorders the see and seealso fields according to the referenced entry’s name (obtained with \glsentryname).

For example, the see field for the duckling entry was originally:

\texttt{see=\{[related terms]fluffy,velociraptor,duck,tardigrade\}}

but in the .glstex file it’s written as:

\texttt{see=\{[related terms]duck,fluffy,velociraptor,tardigrade\}}

The reason for tardigrade being placed after velociraptor is because \glsentryname{tardigrade} is expanded to “water bear” (and “W” comes after “V”). If no encapsulating command was specified:

\texttt{sort-label-list=\{seealso,see\}:en}

then the list would have been sorted according to the labels instead (and so tardigrade would come before velociraptor). Note that the optional tag is kept at the start of the list.

The seealso fields have also been changed. For example, the duck entry originally had:

\texttt{seealso=\{swan,duckling,parrot,goose\}}

but in the .glstex file it’s written as:

\texttt{seealso=\{duckling,goose,parrot,swan\}}

Note that the hierarchical structure hasn’t been maintained. The glossary lists “duckling” (a top-level entry) after “swan” (a level 2 entry) but the seealso field has duckling first.

If you want to maintain the hierarchy you can use \glsxtrhiername instead of \glsentryname:
5.7 Field and Label Options

\GlsXtrLoadResources[
  src={entries},
  sort={en},
  sort-label-list={\seealso,see}:en:glxtriernamesep
]

The separator between the levels is given by \glxtriernamesep which is defined by glossaries-extra to produce “⊿”. The bib2gls interpreter’s definition of this command is different to assist sorting and simply expands to a full stop to prevent it from being replaced by the default word break marker.

In this case \glxtriernamesep{swan} would be displayed as "bird ⊲ waterfowl ⊲ swan" if used in the document, but the interpreter converts it to "bird.waterfowl.swan", so with the default break-at setting the actual sort value becomes bird.waterfowl.swan (instead of bird|waterfowl|swan) which would be the result if the interpreter used the same definition as glossaries-extra).

Therefore the seealso field for the duck entry ends up as:

seealso={parrot,goose,swan,duckling}

Now swan comes before duckling because the actual sort value started with a “B” not “S”.

This hierarchical information isn’t shown in the cross-reference by default, so the duck cross-reference list appears in the document as: parrot, goose, swan & duckling.

If you want the hierarchical information to appear to help assist the reader, you can redefine \glsseeitemformat in the document to use \glxtriernamesep:

\renewcommand*{\glsseeitemformat}[1]{\glxtriernamesep{#1}}

This means that the duck cross-reference now appears in the document as: bird ⊲ parrot, bird ⊲ waterfowl ⊲ goose, bird ⊲ waterfowl ⊲ swan & duckling.

This next example document has two languages, English and Portuguese. The file entries-en.bib contains the English terms, such as:

@index{cat, translations={gato,gatinho} }
@index{kitten, translations={gatinho} }
@index{staple, translations={grampo}}
@index{rivet, translations={rebite}}

The file entries-pt.bib contains the Portuguese terms, such as:

@index{gato, translations={cat,staple,rivet} }
@index{gatinho, translations={kitten} }

Both files have a custom field called translations that will need to be either defined or aliased. This field contains a comma-separated list of labels for the corresponding entries in the other language file that provide a possible translation. Where a word has multiple possible translations, I’d like the list sorted alphabetically. (In practice, it would make more sense
to sort them according to how likely the translation is, but this is for illustrative purposes.)
For convenience, the custom field is simply aliased to the \texttt{user1} field.

The document has two glossaries for each set of terms. The English terms are sorted according to \texttt{sort={en-GB}} in one resource set and the Portuguese terms are sorted according to \texttt{sort={pt-BR}} in another resource set. This means that there are cross-resource references, but since there are no instances of \texttt{@preamble} it should be possible to resolve the references.

The document code is:

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[british,brazilian]{babel}
\usepackage[record,
nomain,
nostyles,
stylemods={bookindex},
style={bookindex}][glossaries-extra]
\usepackage{glossaries-prefix}
\newglossary*[en]{English Terms}
\newglossary*[pt]{Portuguese Terms}

\GlsXtrLoadResources[
  selection={all},
  type={en},
  src={entries-en},
  sort={en-GB},
  field-aliases={translations=user1},
  sort-label-list={user1:pt-BR:glsentryname}
]
\GlsXtrLoadResources[
  selection={all},
  type={pt},
  src={entries-pt},
  sort={pt-BR},
  field-aliases={translations=user1},
  sort-label-list={user1:en-GB:glsentryname}
]
\apptoglossarypreamble[en]{\selectlanguage{british}}
\apptoglossarypreamble[pt]{\selectlanguage{brazilian}}
5.7 Field and Label Options

In verbose mode, the transcript file indicates that it’s performing the label list sorting. For example, when sorting according to `sort-label-list={user1:pt-BR:glsentryname}`, the transcript file contains:

```
Label list sort method 'pt-BR' on field: user1
```

The `cat` entry has a list of two elements in this field: `gato`, `gatinho`. This is converted into a new list where the first element is:

```
{\glsentryname{gato}}
```

and the second element is:

```
{\glsentryname{gatinho}}
```

Regardless of the level of verbosity, the transcript file will contain the conversions obtained by the interpreter:

```
texparserlib: {\glsentryname{gato}} -> gato
texparserlib: {\glsentryname{gatinho}} -> gatinho
```

The `kitten` entry has the same list, and the same process is repeated for that entry. The `--verbose` mode will provide additional information. The `--debug` mode will indicate whether the referenced label was found in the current resource set or if it had to be fetched from another resource set. So if the resulting order isn’t what you expect, check the transcript file for messages.

```
prune-xr=(boolean)
```

If true, this is a shortcut for:

```
prune-see-match={entrytype={index(plural)?},see={},seealso={},alias={}},
prune-seealso-match={entrytype={index(plural)?},see={},seealso={},alias={}},
```

This will remove any labels in an entry’s `see` or `seealso` field where the referenced label doesn’t have any records and hasn’t been selected as another form of dependency and whose entry type is either `@index` or `@indexplural` and doesn’t have the `see`, `seealso` or `alias` fields set.

Both `prune-see-match` and `prune-seealso-match` can be switched off at the same time with `prune-xr={false}`.
5.7 Field and Label Options

prune-see-match={key=value list}

The value has the same syntax as match. Omitting the value switches off the setting. This option is not cumulative.

If a value is supplied, this setting will attempt to prune unnecessary labels from see fields. Note that pruning may fail if there are cross-reference trails.

A label will be stripped from a see field if the label references an entry that has no records, isn’t dependent on another entry, hasn’t previously been selected, and matches the given criteria. If more that one pattern match is supplied, prune-see-op determines whether to apply a logical AND or a logical OR.

For example, suppose the file entries.bib contains the following:

```latex
@index{pumpkin}
@index{cucumber}
@index{melon}
@index{cucurbit, see={gourd}}
@index{gourd, see={pumpkin,cucumber,melon}}
@index{courgette}
@index{marrow, seealso={courgette}}
@index{broccoli}
@index{cauliflower, seealso={broccoli}}
```

Suppose the document contains:

```latex
\GlsXtrLoadResources[src={entries}]
\begin{document}
\gls{cucurbit}, \gls{pumpkin}, \gls{melon}, \gls{broccoli}, \gls{marrow}, \gls{cauliflower}.
\printunsrtglossary[title=Index]
\end{document}
```

This uses the default selection={recorded and deps} setting, which selects recorded entries (cucurbit, pumpkin, melon, broccoli, marrow and cauliflower) and their dependencies. In this case, the dependencies are: courgette (because it’s listed in the marrow’s seealso field), gourd (because it’s listed in the cucurbit’s see field), and cucumber (because it’s listed in the gourd’s see field). The resulting list is:

broccoli 1
cauliflower 1, see also broccoli
courgette
cucumber
cucurbit 1, see gourd
gourd see pumpkin, cucumber & melon
marrow 1, see also courgette
melon 1
pumpkin 1
This means that courgette and cucumber appear in the glossary without a location list. If this was an actual glossary with descriptions, this may not be a problem, but it looks strange for an index since the cross-reference essentially leads the reader to a dead end.

Switching to `selection={recorded no deps}` will remove courgette, gourd and cucumber but the `see` and `seealso` fields will be lost. Since gourd references both pumpkin and melon (which are used in the document), it might be useful to keep the gourd entry. The aim of pruning is to remove the unwanted cucumber entry from the gourd’s `see` list but retain pumpkin and melon.

An appropriate filter is needed to switch on pruning. (This is in addition to the criteria that the pruned entry has no records, isn’t dependent on another entry, and hasn’t previously been selected.) This type of pruning is usually only necessary for indexes so a useful filter may be simply on the entry type (either `@index` or `@indexplural`):

```
\GlsXtrLoadResources[src={entries}, prune-see-match={entrytype={index(plural)?}}]
```

Another possibility is to filter on an empty `description`:

```
\GlsXtrLoadResources[src={entries}, prune-see-match={description={}}]
```

The result is that the cauliflower and marrow entries keep their `seealso` lists (since this option only applies to `see` lists) and the courgette entry has been added (because it’s in the marrow entry’s `seealso` list). The gourd entry is removed from the cucurbit’s `see` list (because it matches the criteria) and is not selected (because it’s no longer a dependency).

In this case, I’d like to include the gourd entry because it has the `see` field set. This means adjusting the criteria so that only entries without the `see` field can be pruned:

```
\GlsXtrLoadResources[src={entries}, prune-see-match={entrytype={index(plural)?},see={}}]
```

This means that gourd is now selected (and retained in the cucurbit’s `see` field) but cucumber is removed from the gourd’s `see` field.

A similar method can be applied for the `seealso` fields using `prune-seealso-match`. There’s no applicable setting for the `alias` field (since it’s expected that the alias be present due to the nature of the way the `alias` field works).

For convenience, the `prune-xr` option is provided as a shortcut. If the resource command in the above example is modified to:

```
\GlsXtrLoadResources[src={entries},prune-xr]
```

then the resulting list will be:

- broccoli 1
- cauliflower 1, see also broccoli
- cucurbit 1, see gourd
- gourd see pumpkin & melon
- marrow 1
- melon 1
- pumpkin 1
Note that if the pumpkin and melon references are removed from the document, then
gourd will still be selected but will have no cross-reference. This is because the cucurbit
entry is checked for pruning while the gourd entry still has a non-empty see field so it’s not
removed from the cucurbit entry.

There are two ways around this problem: either switch the definitions of cucurbit and
gourd around in the .bib file or use prune-iterations to reprune (in this case, prune
-iterations={2} is sufficient).

This setting is only compatible with the “recorded and dep” selection criteria:
selection={recorded and deps}, selection={recorded and deps and see}
and selection={recorded and deps and see not also}.

prune-see-op=(value)
If the value of prune-see-match contains more than one ⟨key⟩=⟨pattern⟩ element, the prune
-see-op determines whether to apply a logical AND or a logical OR. The ⟨value⟩ may be
either and or or. The default is prune-see-op={and}.

prune-seealso-match=(key=value list)
As prune-see-match but for seealso fields. If more that one pattern match is supplied,
prune-see-op determines whether to apply a logical AND or a logical OR.

This setting is only compatible with the “recorded and dep” selection criteria:
selection={recorded and deps}, selection={recorded and deps and see}
and selection={recorded and deps and see not also}.

prune-see-op=(value)
If the value of prune-seealso-match contains more than one ⟨key⟩=⟨pattern⟩ element, the prune
-see-op determines whether to apply a logical AND or a logical OR. The ⟨value⟩ may be
either and or or. The default is prune-see-op={and}.

prune-iterations=(number)
If you have cross-reference trails, you may need to reprune. The value of this options indi-
cates the number of pruning iterations. The default is 1. The higher the number, the longer
tenbib2gls will take to complete. The value can’t be less that 1.
The maximum number of iterations is capped at 20. A cross-reference trail that long is
excessive for an index.
5.7 Field and Label Options

bibtex-contributor-fields={⟨list⟩}

This option indicates that the listed fields all use \textsc{Bib}TeX’s name syntax (as used in \textsc{Bib}TeX’s author and editor fields). The value is required for this key but may be empty, which indicates an empty set of fields (that is, the setting is switched off).

The values of these fields will be converted into the form:

\texttt{\textbackslash bibglstcontributorlist\{⟨contributor list⟩\}\{⟨n⟩\}}

where ⟨⟨n⟩⟩ is the number of names in the list and ⟨⟨contributor-list⟩⟩ is a comma-separated list of names in the form:

\texttt{\textbackslash bibglstcontributor\{⟨forenames⟩\}\{⟨von-part⟩\}\{⟨surname⟩\}\{⟨suffix⟩\}}

The \texttt{\textbackslash bibglstcontributorlist} command is initially defined in \texttt{bib2gls}’s interpreter to just do the first argument and ignore the second. This means that if you’re sorting on this field, the “and” part between the final names doesn’t appear in the sort value. The actual definition of \texttt{\textbackslash bibglstcontributorlist} provided in the .glstex file depends on whether or not \texttt{\textbackslash DTLformatlist} is defined. (Note that glossaries automatically loads datatool-base so this command will be defined if you have at least v2.28 of datatool-base.)

For example, if the name field is specified as:

\texttt{name={John Smith and Jane Doe and Dickie von Duck}}

then \texttt{bibtex-contributor-fields={name}} will convert the name field value to:

\texttt{\textbackslash bibglstcontributorlist\{%
\textbackslash bibglstcontributor\{John\}\{Smith\}\{,\}
\textbackslash bibglstcontributor\{Jane\}\{Doe\}\{,\}
\textbackslash bibglstcontributor\{Dickie\}\{von\}\{Duck\}\{\}\{3\}}

With \texttt{contributor-order={von}} the sort value obtained from this field will be:

Smith, John, Doe, Jane, von Duck, Dickie

With one of the locale sort methods and with the default \texttt{break-at={word}}, this will end up as:

Smith|John|Doe|Jane|von|Duck|Dickie

\texttt{contributor-order={⟨value⟩}}

The \texttt{\textbackslash bibglstcontributor} command is defined in \texttt{bib2gls}’s interpreter and its definition is dependent on this setting. The ⟨⟨value⟩⟩ may be one of (where the parts in square brackets are omitted if that argument is empty):

- \texttt{surname}: \texttt{\textbackslash bibglstcontributor} expands to \langle surname ⟩[. ⟨suffix ⟩][. ⟨forenames ⟩][. ⟨von-part ⟩];
5.7 Field and Label Options

- von: \bibglscontributor expands to \((\text{\textit{von-part}}) \text{\textit{surname}}, \text{\textit{suffix}})\;\text{\textit{forenames}};

- forenames: \bibglscontributor expands to \((\text{\textit{forenames}}) [(\text{\textit{von-part}}) \text{\textit{surname}} [\text{\textit{suffix}}]) [\text{\textit{forenames}}]]\).

The default value is von. Note that if you have multiple resource sets, this option governs the way \bib2gls's version of \bibglscontributor behaves. The actual definition is written to the .glstex using \providecommand, which means that \LaTeX{} will only pick up the first definition.

For example:

\begin{verbatim}
\newcommand*{\bibglscontributor}[4]{%
  #1\ifstrempty{#2}{}{ #2} #3\ifstrempty{#4}{}{, #4}%
}
\end{verbatim}

\begin{verbatim}
\GlsXtrLoadResources[  
  src={entries},% data in entries.bib  
  bibtex-contributor-fields={name}  
]
\end{verbatim}

This will display the names in the glossary with the forenames first, but \bib2gls will sort according to surname.

An alternative approach, if you need an initial resource set such as with the no-interpret-preamble.bib file:

\begin{verbatim}
\GlsXtrLoadResources[  
  src={no-interpret-preamble},  
  interpret-preamble={false},  
  bibtex-contributor-fields={name},  
  contributor-order={forenames}  
]
\end{verbatim}

\begin{verbatim}
\GlsXtrLoadResources[  
  src={entries},% data in entries.bib  
  bibtex-contributor-fields={name}  
]
\end{verbatim}

Note the need to use \texttt{bibtex-contributor-fields={name}} in the first resource set even though there are no entries in the .bib file. This is because the definition of \texttt{\bibglscontributor} is only written to the .glstex file if \texttt{bibtex-contributor-fields} has been set to a non-empty list. The second resource set will use the default \texttt{bibtex-contributor-fields={von}} setting when obtaining the sort value.

\begin{verbatim}
encapsulate-fields=\{(key=value list)\}
\end{verbatim}

This option should take a comma-separated list of \texttt{(field)=\texttt{cs-name-1arg}} values, where \texttt{(cs-name-1arg)} is the name of a control sequence that takes one argument. The value is required
5.7 Field and Label Options

for this key but may be empty, which indicates an empty set (that is, the setting is switched off).

During the processing stage, each field identified in the list (if defined) will have its value replaced with:

\(\langle cs-name-1arg\rangle\{\langle value\rangle\}\)

where \(\langle value\rangle\) was its previous value. An empty list switches off encapsulation (the default).

This action overrides any previous use of \texttt{encapsulate-fields} within the same resource set and is always performed before \texttt{encapsulate-fields*}, regardless of the order in the resource set’s list of options.

\texttt{encapsulate-fields*={\langle key=value list\rangle}}

This option should take a comma-separated list of \langle field\rangle=\langle cs-name-2arg\rangle values, where \langle cs-name-2arg\rangle is the name of a control sequence that takes two arguments. The value is required for this key but may be empty, which indicates an empty set (that is, the setting is switched off).

During the processing stage, each field identified in the list (if defined) will have its value replaced with:

\(\langle cs-name-2arg\rangle\{\langle value\rangle\}\{\langle label\rangle\}\)

where \(\langle value\rangle\) was its previous value and \(\langle label\rangle\) is the entry’s label (including prefix, if appropriate). An empty list switches off encapsulation (the default).

This action overrides any previous use of \texttt{encapsulate-fields*} within the same resource set, and is always performed after \texttt{encapsulate-fields}, regardless of the order in the resource set’s list of options, so if the same field is listed in both settings, its value will end up as:

\(\langle cs-name-2arg\rangle\{\langle cs-name-1arg\rangle\{\langle value\rangle\}\}\{\langle label\rangle\}\)

\texttt{format-integer-fields={\langle key=value list\rangle}}

This option should take a comma-separated list of \langle field\rangle=\langle format\rangle values, where \langle format\rangle is a string format pattern that contains a single numeric specifier. This will convert the value stored in the identified field to the given format. If the field doesn’t contain an integer value it won’t be changed. If the field contains a decimal value use \texttt{format-decimal-fields} instead. This setting is performed before field encapsulation.

Since format patterns uses % as a placeholder, which can be problematic in the resource command, you will need to use \texttt{\%\%} instead. You may also use \texttt{\#, \$, \&, \-, \_ and \\} to indicate the corresponding literal character. You can use \texttt{\uXXXX} to indicate a character by its hexadecimal value, but remember that the resource options will be expanded when they are written to the resource file so use \texttt{\glshex or \string\u}.

If you want to format the \texttt{definition-index} field you must use \texttt{save-definition-index} first. For example, to save this field and then zero-pad it to four digits:
5.7 Field and Label Options

save-definition-index,
format-integer-fields={definitionindex=\%04d}

This option can’t be used for the useindex field created with save-use-index as that field isn’t set until after the field modifications are made.

format-decimal-fields={{key=value list}}

As format-integer-fields but for decimal values. If a field contains an integer then:

• if format-integer-fields has also been used to set a format for the given field, the integer format will take precedence;

• otherwise the integer value will be treated as a decimal number.

If you get an error like:

Error: d != java.lang.Double

then it means you have used an invalid specifier. (The above error results from using %d instead of %f or %g.)

interpret-fields={{list}}

This option indicates that the listed fields should be replaced by their interpreted values. The value is required for this key but may be empty, which indicates an empty set of fields (that is, the setting is switched off). Other fields not listed may still be interpreted depending on other settings. As with the sort field, any special characters are replaced with commands like \glsbackslash and \bibglsdollarchar. This option is applied after field-case-change (if set).

For example, suppose I have a file entries.bib that contains definitions like:

@symbol{pi,
  name={\ensuremath{\pi}},
  description={the ratio of a circle's circumference to its diameter},
}

@symboll{sigma,
  name = {\ensuremath{\sigma}},
  description = {standard deviation}
}

Instead of having a list of terms (glossary), suppose I want to have stand-alone definitions, where the term appears in a section heading. I could define a command like this:
5.7 Field and Label Options

\newcommand{\definition}[1]{\%}
\ifglsentryexists{#1}{%
  \section{\glsentryname{#1}}{\glsadd{#1}\glsxtrglossentry{#1}}%
  \Glossentrydesc{#1}\glspostdescription
\}%
\{\section{Missing `#1']}{\glsadd{#1}}}%
}

which can be used in the document:

\tableofcontents
\definition{pi}
\definition{sigma}

A problem with this definition of my custom command occurs if I add \texttt{hyperref} to the document, because this tries to write $\pi$ and $\sigma$ to the PDF bookmarks, which doesn’t work because those commands can’t be automatically converted to characters permitted in a PDF string. This leads to a warning from \texttt{hyperref}:

\textbf{Token not allowed in a PDF string (Unicode)}

Ideally I’d like to be able to convert these symbols to Unicode so that they can appear in the bookmarks. Since \texttt{bib2gls’} interpreter recognises these commands, I can get it to make the conversion instead of trying to implement a method within \LaTeX:

\glsaddstoragekey{pdfname}{}{\pdfname}
\GlsXtrLoadResources[  src={entries},  replicate-fields={name=pdfname},  replicate-missing-field-action={fallback},  interpret-fields={pdfname} ]

This first copies the \texttt{name} field to the custom \texttt{pdfname} and then interprets the copy. This leaves the \texttt{name} field with the \LaTeX code to produce the symbol in the document, but the \texttt{pdfname} field ends up with all markup stripped by the interpreter and the $\pi$ and $\sigma$ are converted to the Unicode characters 0x1D70B (mathematical italic small pi) and 0x1D70E (mathematical italic small sigma). With \texttt{XeLaTeX} or \texttt{LuaLaTeX} these characters can be written to the PDF bookmarks by adjusting the definition of the custom command:

\newcommand{\definition}[1]{\%}
\ifglsentryexists{#1}{%
  \section{\texorpdfstring{\glsentryname{#1}}{\pdfname{#1}}}{\glsadd{#1}\glsxtrglossentry{#1}}%
  \Glossentrydesc{#1}\glspostdescription
  {\pdfname{#1}}\glsxtrglossentry{#1}}%
}

\newpage
5.7 Field and Label Options

\Glossentrydesc{#1}\glspostdescription
\%
{\section[Missing \#1]{\glsadd{#1}}}\}

With pdf\LaTeX\ and fontenc, you will need hyperref’s \texttt{unicode} option:

\usepackage[unicode]{hyperref}

If you still encounter problems with the Unicode characters not appearing in the PDF bookmarks, then try the \texttt{hex-unicode-fields} option. For example:

\texttt{hex-unicode-fields=\{pdfname\}}

This still requires hyperref’s \texttt{unicode} option.

\texttt{interpret-fields-action=\{\langle value\rangle\}}

This option governs the behaviour of \texttt{interpret-fields}. Available values are:

- \texttt{replace}: replace the field content with its interpreted value (default);
- \texttt{replace non empty}: only replace the field content with its interpreted value if the interpreted value isn’t an empty string.

If a field value consists solely of commands that are unknown to the interpreter, then the resulting value will end up empty. In this case, it may be more appropriate to leave the field unchanged.

\texttt{hex-unicode-fields=\{\langle list\rangle\}}

This option will convert any Unicode characters (outside of the Basic Latin set) that are found in the listed fields into \texttt{\bibglshexunicodechar{\langle hex-code\rangle}} where \texttt{\langle hex-code\rangle} is the hexadecimal character code.

The \texttt{\langle list\rangle} should be a comma-separated list of field names. This action is performed after \texttt{interpret-fields}.

If the field contents need to be added to the PDF bookmarks (as in the earlier example) then you need to make sure you use hyperref’s \texttt{unicode} option otherwise you’ll get the warning:

Token not allowed in a PDF string (PDFDocEncoding):
removing `\char'

and the bookmarks will show \texttt{\langle hex-code\rangle} instead of the Unicode character.
5.7 Field and Label Options

date-time-fields=⟨list⟩

This option indicates that the listed fields all contain date and time information. Primary entries will have these fields parsed according to date-time-field-format and date-time-field-locale and dual entries will have these fields parsed according to dual-date-time-field-format and dual-date-time-field-locale. If the field value is missing or doesn’t match the given pattern it remains unchanged, otherwise it’s converted into the form:

\bibglsdatetime{⟨year⟩}{⟨month⟩}{⟨day-of-month⟩}{⟨day-of-week⟩}{⟨day-of-year⟩}
{⟨era⟩}{⟨hour⟩}{⟨minute⟩}{⟨second⟩}{⟨millisec⟩}{⟨dst⟩}{⟨zone⟩}{⟨original⟩}

where ⟨original⟩ is the value of the field before conversion. If the interpreter is on, the value will be interpreted before being parsed if it contains \, $, {, } or ~. (Remember that ~ is converted to the non-breaking space character 0xA0 unless --break-space is used.)

date-fields=⟨list⟩

As date-time-fields but for fields that only contain date (not time) information. If parsed correctly, the field is converted to:

\bibglstime{⟨year⟩}{⟨month⟩}{⟨day-of-month⟩}{⟨day-of-week⟩}{⟨day-of-year⟩}
{⟨era⟩}{⟨original⟩}

The fields are parsed according to date-field-format and date-field-locale for primary entries and according to dual-date-field-format and dual-date-field-locale for dual entries.

time-fields=⟨list⟩

As date-time-fields but for fields that only contain time (not date) information. If parsed correctly, the field is converted to:

\bibglstime{⟨hour⟩}{⟨minute⟩}{⟨second⟩}{⟨millisec⟩}{⟨dst⟩}{⟨zone⟩}{⟨original⟩}

The fields are parsed according to time-field-format and time-field-locale for primary entries and according to dual-time-field-format and date-time-field-locale for dual entries.

date-time-field-format=⟨value⟩

This option also sets dual-date-time-field-format=⟨value⟩. The value is the format pattern used when parsing fields identified by date-time-fields. The ⟨value⟩ is as for date-sort-format.
date-field-format={value}

This option also sets dual-date-field-format={value}. The value is the format pattern used when parsing fields identified by date-fields. The (value) is as for date-sort-format.

time-field-format={value}

This option also sets dual-time-field-format={value}. The value is the format pattern used when parsing fields identified by time-fields. The (value) is as for date-sort-format.

date-time-field-locale={value}

This option also sets dual-date-time-field-locale={value}. The value is the locale used when parsing fields identified by date-time-fields. The (value) is as for date-sort-locale.

date-field-locale={value}

This option also sets dual-date-field-locale={value}. The value is the locale used when parsing fields identified by date-fields. The (value) is as for date-sort-locale.

time-field-locale={value}

This option also sets dual-date-time-field-locale={value}. The value is the locale used when parsing fields identified by time-fields. The (value) is as for date-sort-locale.

Prefix Fields

If you use the glossaries-prefix package, the prefix set of fields become available (prefix, prefixplural, prefixfirst and prefixfirstplural). The default behaviour of \gls is for no separator between the prefix and the text produced with \gls. This is because there are situations where there shouldn’t be a space, although a space is more commonly required.

This means that a space needs to be appended to the required prefix fields, but an actual space character can’t be used because \xkeyval trims leading and trailing spaces. The \space command needs to be used instead, but there are also situations where an non-breakable space should be used (for example, where the prefix is a single character). It’s a bit tiresome having to remember to put \space or ~ at the end of the field value.

The append-prefix-field option allows the automatic insertion of a space, but it may be used without the glossaries-prefix package. The fields that contain prefixes are identified by prefix-fields.

If you have any dual entries, then \bib2gls will also recognise the special internal fields dualprefix, dualprefixplural, dualprefixfirst and dualprefixfirstplural.
5.7 Field and Label Options

prefix-fields=⟨list⟩
Identifies the fields that are used to store prefixes. The default set is: prefix, prefixfirst, prefixplural, prefixfirstplural, and their dual counterparts dualprefix, dualprefixfirst, dualprefixplural and dualprefixfirstplural.

append-prefix-field=⟨value⟩
Allowed values are:

- none: don’t append a space to the prefix fields (default);
- space: append the command identified by append-prefix-field-cs (\space by default) to the prefix field unless the field value ends with a character identified by append-prefix-field-exceptions or a command identified by append-prefix-field-cs-exceptions. Note that if the field value ends with anything else (such as an empty group) then these exceptions won’t apply.
- space or nbsp: as above but uses ~ instead of \space if the field value matches the pattern given by append-prefix-field-nbsp-match.

append-prefix-field-cs=⟨cs⟩
Identifies the command ⟨cs⟩ that should be used to append to the prefix fields. The default value is \space. Remember to use \string or \protect to prevent the command from being expanded as it’s written to the .aux file.

append-prefix-field-exceptions=⟨sequence⟩
This setting identifies the set of characters that, if found at the end of a prefix field, prevent append-prefix-field from appending a space (either \space or ~).

The value should be a sequence of characters. You may use \string\u⟨hex⟩ to identify a character by its hexadecimal code. Spaces are ignored, so append-prefix-field-exceptions={' - } is equivalent to append-prefix-field-exceptions={'-}.

The default set is the straight apostrophe character (0x0027), the hyphen-minus character (0x002D), the tilde character (~), the hyphen character (0x2010), the non-breaking hyphen (0x2011), and the right single quotation mark (0x2019).

append-prefix-field-cs-exceptions=⟨sequence⟩
This setting identifies the set of commands that, if found at the end of a prefix field, prevent append-prefix-field from appending a space (either \space or ~). Any spaces found in ⟨sequence⟩ are ignored. The default setting is the set: \space, \nobreakspace and \␣.

Remember that you will need to use \string or \protect to prevent the command from being expanded while the resource options are written to the .aux file.
5.7 Field and Label Options

append-prefix-field-nbsp-match={pattern}

The value is the regular expression that identifies prefixes that should be followed by ~ instead of \space. The default is append-prefix-field-nbsp-match={.} which indicates a single character.

Case-Changing

The glossaries-extra package comes with the category attributes glossdesc and glossname, which may take the values firstuc or title. These don’t change the actual name or description fields, but instead \glossentryname and \glossentrydesc (which are used by the default glossary styles) check for the corresponding attribute and apply the appropriate case-change to the field value.

So \glossentryname will use \Glsentryname if the glossname attribute for the given entry is set to firstuc and \glossentrydesc will use \Glsentrydesc if the glossdesc attribute is set to firstuc. The title setting will instead use \capitalisewords applied to the field value.

The resource options described in this section provide an alternative to those attributes that actually modify the relevant field (rather than just adjusting the style code used to display it). There are two forms of modification: the field is adjusted so that the original value is encapsulated by a command or bib2gls will perform the actual case-change according to its own algorithm. The results can vary according to the field content.

Where bib2gls itself performs the case change, its case-changing functions will use the resource locale, but whether or not bib2gls recognises the correct rules for the locale depends on whether or not the locale is correctly supported by the Java locale provider. The language resource file may provide assistance with case-conversion.

Each of the case-changing resource options may take one of the following values:

• none: don’t apply any case-changing (default);
• lc-cs: make bib2gls behave as though the field assignment:

\langle field \rangle = \{\langle text \rangle\}

had actually been specified as:

\langle field \rangle = \{\bibglslowercase{\langle text \rangle}\}

which uses \TeX to convert the field to lower case;
• uc-cs: make bib2gls behave as though the field assignment:

\langle field \rangle = \{\langle text \rangle\}

had actually been specified as:

\langle field \rangle = \{\bibglsuppercase{\langle text \rangle}\}
which uses \TeX\ to convert the field to upper case;

- \texttt{firstuc-cs}: make \bib2gls\ behave as though the field assignment:

\begin{verbatim}
\langle field \rangle = \{\langle text \rangle\}
\end{verbatim}

had actually been specified as:

\begin{verbatim}
\langle field \rangle = \{\\bibglsfirstuc\{\langle text \rangle\}\}
\end{verbatim}

which uses \TeX\ to convert the field to first-letter upper case;

- \texttt{title-cs}: make \bib2gls\ behave as though the field assignment:

\begin{verbatim}
\langle field \rangle = \{\langle text \rangle\}
\end{verbatim}

had actually been specified as:

\begin{verbatim}
\langle field \rangle = \{\\bibglstitlcase\{\langle text \rangle\}\}
\end{verbatim}

which uses \TeX\ to convert the field to title case;

- \texttt{lc}: convert to lower case by making the appropriate modifications to tokens in the field value that have a known lower case alternative (see below);

- \texttt{uc}: convert to upper case by making the appropriate modifications to tokens in the field value that have a known upper case alternative (see below);

- \texttt{firstuc}: convert to first letter upper case by making the appropriate modification, if it has a known upper case alternative (see below);

- \texttt{title}: convert to title case by making the appropriate modifications to the first letter of each identified word in the field value that has a known upper case alternative (see below).

A word-boundary is identified according to the \texttt{word-boundaries} setting. Words to be excluded from the case-changing (unless they occur at the start) can be identified with \texttt{\MFUnocap} in the \texttt{@preamble} or you can use \texttt{--packages mfirstuc-english} for the exclusion list provided by the mfirstuc-english package. Alternatively, you can use \texttt{--custom-packages} to load a simple package that contains the required \texttt{\MFUno-cap} commands (in a similar style to mfirstuc-english).

The \bib2gls\ word-boundary implementation is slightly different with this setting than with the \texttt{\capitalisewords} command (implemented in \TeX\ or by the \TeX\ parser library when interpreting field values). Only words in the exclusion list that start with an alphabetical character can be matched. Punctuation following a word-boundary is not considered part of the next word. If you want to identify that a particular character forms a word break, you can use \texttt{\MFUwordbreak\{\langle char \rangle\}}. For example:
5.7 Field and Label Options

name={some word\MFUwordbreak{/}phrase}

If you need to selectively change the case, based on some condition (such as the entry type) then you can use the assign-fields option instead, but remember that you will need the override setting on. For example:

assign-fields={
name =[@] \TITLE{ name }
[ entrytype -> original = "entry" ]
}

This will convert the *name* field to title case for entries that were defined in the .bib file with @entry. Note that if you also use a case-changing option, for example, name-case-change, then all entries will have the change applied, according to the option’s designated behaviour, regardless of whether or not the applicable field has already been altered by assign-fields.

Major changes have been introduced to mfirstuc v2.08. Some of the information below refers to older versions and is not applicable with mfirstuc v2.08+. See the mfirstuc manual for further details.

The firstuc-cs and firstuc options are essentially a sentence case change, but there’s no check for sentence-breaks within the value, so even if the value contains multiple sentences, only the first is changed. If the text to be changed starts with a punctuation character it should be encapsulated with \MFUskippunc to apply the case-change to the following object. For example:

name={\MFUskippunc{'}tis}

If the firstuc option is applied to the *name* field this will be converted to:

name={\MFUskippunc{'}Tis}

Using \NoCaseChange (provided by textcase) instead will have the same effect, but this isn’t consistent with the behaviour of \makefirstuc so it’s best to use \MFUskippunc instead.

The ⟨option⟩-cs settings defer the actual case-changing to \TeX, which means that the case-changing has to be applied every time the field is typeset (and it introduces non-expandable content to the field value). Be aware of the limitations of using any of the case-changing commands. See the textcase and mfirstuc package documentation for further details [1, 11].

For the settings where bib2gls itself performs the case-change, then bib2gls will iterate over each token of the field value and apply the rules listed below. Note that the case-change implemented by bib2gls recognises the resource locale, but whether or not it recognises the correct rules for the locale depends on whether or not the locale is correctly supported by the Java locale provider.

1. If the token is a normal Unicode alphabetic character, it will be replaced with the corresponding upper or lower case character, as appropriate. In some cases, a single character, such as ß, is replaced by multiple characters, such as SS.

208
For title and firstuc, the title case character is used as the replacement, for uc the upper case character is used as the replacement, and for lc the lower case character is used as the replacement. Many characters have the same upper and title case alternative (for example, “a” will be converted to “A” for the title, firstuc and uc settings), but some characters have different title and upper versions (for example, the digraph “dz” has the title version “Dz” and upper case version “DZ”).

If the option is firstuc then all the remaining tokens are skipped. If the option is title then the subsequent tokens are skipped until a word-boundary is found.

2. If the token is a normal Unicode character that isn’t alphabetical, then this token will be skipped for all options.

3. If \$\langle maths\rangle\$ is encountered, it will be skipped. If the option is firstuc then all remaining tokens are skipped, so no case-change will be performed.

4. If a group \{\langle content\rangle\} is found, then the case-change is applied to the entire \langle content\rangle (which may be empty). This corresponds to the way \texttt{\textbackslash makefirstuc} and \texttt{\textbackslash capitalise-words} work if a word starts with a group. Note that with firstuc and title the group content will be converted according to uc, so the normal upper case character is used rather than the title case character (if they are different).

If the option is firstuc then all the remaining tokens are skipped. If the option is title then the subsequent tokens are skipped until a word-boundary is found. A word-boundary can be marked up with \texttt{\textbackslash MFU\textbackslash wordbreak}.

5. If a control sequence \texttt{\langle csname\rangle} is found, then:
   a) If the control sequence is \texttt{\textbackslash protect}, this token is skipped for all options.
   b) With firstuc and title, if \texttt{\textbackslash MFU\textbackslash skippunc\{\langle text\rangle\}} or \texttt{\textbackslash NoCaseChange\{\langle text\rangle\}} occurs at the start of a word, then bib2gls will act as though the word hasn’t started yet (so the next token will be considered for a case-change).
   c) If the control sequence is one of: \\o, \O, \l, \L, \ae, \AE, \oe, \OE, \aa, \AA, \ss, \SS, \ng, \NG, \th, \TH, \dh, \DH, \dj or \DJ, then it’s replaced with its case-change counterpart (if not already the correct case).

If the option is firstuc then all the remaining tokens are skipped. If the option is title then the subsequent tokens are skipped until a word-boundary is found.

If the option is firstuc then all the remaining tokens are skipped (so no case-change will be performed). If the option is title then the subsequent tokens are
skipped until a word-boundary is found (so no case-change is performed for this word).

e) If the control sequence is \texttt{\glsentrytitlecase} then:
\begin{itemize}
  \item \texttt{lc} the control sequence is converted to \texttt{\glsxtrusefield};
  \item \texttt{uc} the control sequence is converted to \texttt{\GLSxtrusefield};
  \item \texttt{firstuc} the control sequence is converted to \texttt{\Glsxtrusefield} and the remaining tokens are skipped;
  \item \texttt{title} the control sequence is left unchanged and subsequent tokens are skipped until a word-boundary is found.
\end{itemize}

The field and entry label arguments are skipped.

f) If the control sequence is \texttt{\glshyperlink} then the case-change is applied to its optional argument. (If there was no optional argument in the original field value, one will be inserted.) The label argument is skipped.

   If the option is \texttt{firstuc} then all the remaining tokens are skipped. If the option is \texttt{title} then the subsequent tokens are skipped until a word-boundary is found.

g) If the control sequence is \texttt{\glsdisp}, \texttt{\glslink}, \texttt{\dglsdisp} or \texttt{\dglslink} then the case-change will be applied to the appropriate argument. The optional argument (if present) and the label are skipped.

   If the option is \texttt{firstuc} then all the remaining tokens are skipped. If the option is \texttt{title} then the subsequent tokens are skipped until a word-boundary is found.

h) If the control sequence has a known case variant, it will be substituted. For example, \texttt{\gl} will be changed to \texttt{\Gls} or \texttt{\GLS}. In some cases there isn’t an appropriate variant. For example, \texttt{\glsentrytext} has a first-letter upper case version \texttt{\Glsentrytext}, but not an all-caps version.

   If the option is \texttt{firstuc} then all the remaining tokens are skipped. If the option is \texttt{title} then the subsequent tokens are skipped until a word-boundary is found.

i) If the control sequence is followed by a group, then the appropriate case-change is applied to the group contents. Unlike step 4, the case-change isn’t applied to the entire group content with \texttt{firstuc} and \texttt{title}. (Again, this follows the way that \texttt{\makefirstuc} and \texttt{\capitalisewords} work.)

   If there are subsequent groups, they won’t be considered arguments, but will be treated as groups, as per step 4. (This will only affect the \texttt{title} setting as they will be skipped by the \texttt{firstuc} setting.) For complex cases, consider using a semantic command that hides non-textual context such as the \texttt{\strong} example described on page 112.

j) Otherwise the control sequence is skipped.

6. Anything else is skipped.
For example, if an entry is defined as:

```latex
@abbreviation{html,
  short = {HTML},
  long = {hypertext markup language},
  description={a markup language for creating web pages}
}
```

then:

```latex
\GlsXtrLoadResources[
  short-case-change={lc},
  long-case-change={title},
  description-case-change={firstuc}
]
```

will make the entry behave as if it had been defined as:

```latex
@abbreviation{html,
  short = {html},
  long = {Hypertext Markup Language},
  description={A markup language for creating web pages}
}
```

whereas:

```latex
\GlsXtrLoadResources[
  short-case-change={lc-cs},
  long-case-change={title-cs},
  description-case-change={firstuc-cs}
]
```

will make the entry behave as if it had been defined as:

```latex
@abbreviation{html,
  short = {\bibglslowercase{HTML}},
  long = {\bibglstitlecase{hypertext markup language}},
  description={\bibglsfirstuc{a markup language for creating web pages}}
}
```

If the given field is missing, no change is made, except under certain circumstances (see the relevant resource option for details). For example, if an abbreviation is simply defined as:

```latex
@abbreviation{html,
  short = {html},
  long = {hypertext markup language}
}
```
then:

\GlsXtrLoadResources[
  name-case-change={uc},
  description-case-change={title}
]

won't have an effect. Although the default long–short abbreviation style sets the name and description fields, bib2gls doesn't have access to this information.

Remember that you can create missing fields by copying the value from another field. So if the resource options are changed to:

\GlsXtrLoadResources[
  name-case-change={uc},
  description-case-change={title},
  replicate-fields={short=name,long=description}
]

then bib2gls will act as though the entry had been defined as:

@abbreviation{html,
  short = {html},
  long = {hypertext markup language},
  name = {HTML},
  description = {Hypertext Markup Language}
}

If the long–short–sc abbreviation style is set (before \GlsXtrLoadResources) then this will override the default style for the name and description, so \gls{html} will display the short form using \textsc{html} but the name in the glossary will be displayed using just HTML.

Note that with @index the name and text fields will automatically be created if they are missing and name-case-change is used. For example, if an entry is defined as:

@index{duck}

then name-case-change={firstuc} will make this entry behave as though it was defined as:

@index{duck,
  name = {Duck},
  text = {duck}
}

Suppose I have a slightly eccentric abbreviation definition:

@abbreviation{html,
  short = "ht\textbf{ml}",
  long = "hypertext markup language"
}
5.7 Field and Label Options

then \texttt{short-case-change=\{uc\}} would convert the value of the \texttt{short} field into:

\texttt{HT\texttt{\emph{ML}}}

Note that \texttt{\emph{}} isn’t modified as it’s recognised as a command. There’s a difference between a group that follows a control sequence and one that doesn’t. For example:

\begin{verbatim}
@abbreviation{html,
    short = "{ht}ml",
    long = "hypertext markup language"
}
\end{verbatim}

In this case \texttt{short-case-change=\{firstuc\}} will convert the \texttt{short} field value to:

\texttt{\{HT\}ml}

(The entire contents of the group \{ht\} has been converted.) Whereas with:

\begin{verbatim}
@abbreviation{html,
    short = "\emph{ht}ml",
    long = "hypertext markup language"
}
\end{verbatim}

then \texttt{short-case-change=\{firstuc\}} will convert the \texttt{short} field value to:

\texttt{\emph{Ht}ml}

(Only the first letter of the argument \{ht\} has been converted.)

There’s no attempt at interpreting the field contents at this point (but the value may later be interpreted during sorting). For example, suppose a \texttt{name} field is defined using:

\begin{verbatim}
name = "z\ae\oe",
\end{verbatim}

then with \texttt{name-case-change=\{uc\}}, the value would be converted to

\texttt{Z\AE\OE}

because \texttt{\ae} and \texttt{\oe} have known upper case versions.

With \texttt{name-case-change=\{uc-cs\}}, the \texttt{name} value would be converted to:

\texttt{\bibglsuppercase{z\ae\oe}}

If the interpreter is used during sorting, the sort value will be set to \texttt{Z\AE\OE} because the interpreter recognises all three commands.

You can use \texttt{\NoCaseChange{\{text\}}} to prevent the given \texttt{\{text\}} from having the case changed. For example, if the \texttt{short} field is defined as:

\begin{verbatim}
short = \{a\NoCaseChange{bc}d\}
\end{verbatim}

then with \texttt{short-case-change=\{uc\}}, this would be converted to
5.7 Field and Label Options

\NoCaseChange{bc}

Note that with \firstuc and title, if \MFUskippunc{(text)} occurs at the start of a word then it’s skipped, and the case change is applied to the material following its argument. For example, suppose the \texttt{short} field is defined as:

\texttt{short={\MFUskippunc{html}}}

then the result is:

\MFUskippunc{h}Tml

whereas with:

\texttt{short={{}html}}

then the result is just {html} (since the case change is applied to the empty group, which has no effect).

If you have a command that takes a label or identifier as an argument then it’s best to hide the label in a custom command. For example, if the \texttt{short} field in the .bib definition is defined as:

\texttt{short = "ht\textcolor{red}{html}"},

then with \texttt{short-case-change={uc}} this would end up as:

HT\textcolor{RED}{ML}

which is incorrect. Instead, provide a command that hides the label (such as the \texttt{\strong} example described on page 112).

\texttt{no-case-change-cs=⟨list⟩}

Instructs the non-TEX case-changing options (where \bib2gls, not TEX, performs the modification) to treat the commands whose control sequence names are given in the comma-separated ⟨list⟩ in the same way as it treats \\ensuremath etc. That is, the case-change is omitted for the argument that follows any of those commands.

For example, this manual defines some semantic commands such as \texttt{\fieldfmt} (to format field names), \texttt{\abbrstylefmt} (to format abbreviation style names) and \texttt{\glostylefmt} (to format glossary style names). If any these occur in section and subsection headings (which are converted to title case) then the case-change would produce an inappropriate result. These formatting commands shouldn’t have their argument changed so they are identified with:

\texttt{no-case-change-cs={fieldfmt,abbrstylefmt,glostylefmt}}
5.7 Field and Label Options

word-boundaries=⟨list⟩

Governs how the title case-change option determines word boundaries. The ⟨list⟩ must contain one or more of the following keywords:

- **white space** any white space Unicode character that is not a non-breakable space indicates a word-boundary;
- **cs space** the control sequences \space or \U indicate a word-boundary;
- **dash** a Unicode character that belongs to the “Punctuation, Dash” block indicates a word-boundary;
- **nbsp** the - active character or the Unicode non-breakable characters 0x00A0, 0x2007 and 0x202F indicate a word-boundary.

Any keyword that is not listed indicates that particular setting is off. This option is not cumulative. Any subsequent use of word-boundaries within the same set of resource options will override previous settings.

The default setting is word-boundaries={white space, cs space}, which excludes non-breakable spaces and dashes.

Note that you can explicitly markup word-boundary punctuation using \MFUwordbreak. For example:

\texttt{name = \{a book of rhyme\MFUwordbreak{/}verse\}}

short-case-change=⟨value⟩

Applies a case-change to the short field (if present). This option may take one of the values described above.

See dual-short-case-change to adjust the dualshort field.

long-case-change=⟨value⟩

Applies a case-change to the long field (if present). This option may take one of the values described above.

See dual-long-case-change to adjust the duallong field.

name-case-change=⟨value⟩

Applies a case-change to the name field. This option may take one of the values described above.

If the text field hasn’t been set, the name value is first copied to the text field. If the name field hasn’t been set (for example, with the @index entry type), it’s copied from the fallback value (which depends on the entry type, see section 5.8) unless the entry type is @abbreviation or @acronym, in which case if the name field is missing no action is performed.
5.8 Field Fallbacks

description-case-change={value}

Applies a case-change to the description field (if present). This option may take one of the values described above.

field-case-change={key=value list}

A general case-change instruction. The value should be a comma-separated list of \langle field\rangle = \langle setting\rangle for each field that needs a case-change applied. The value is required for this key but may be empty, which indicates this option is switched off.

The \langle setting\rangle should be the same as the permitted values for the above options. This option is applied after all fields have been parsed but before interpret-fields. If the specified field is missing, the fallback for that field (if known, see section 5.8) is copied into the field. For example:

field-case-change={user1=uc,user2=firstuc}

This manual provides a custom storage key called nametitle:

\glsxtrprovidestoragekey{nametitle}{\{}{\{}\}

The resource options copy the name value to this custom field and convert nametitle to title case:

replicate-fields={name=nametitle},
field-case-change={nametitle=title},

This means that it’s possible to fetch the value of nametitle instead of name, which provides an expandable title case form that’s suitable for the PDF bookmarks. (Note that \LaTeX{} now provides some expandable case-changing commands.)

This option isn’t cumulative. If used multiple times in the same resource set, the last instance will be the one used. If the key=value list is missing, no general case-changing is applied (the default).

5.8 Field Fallbacks

The options in this section don’t modify any field values but provide instructions on what to do if bib2gls wants to know the value of a field where the field hasn’t been explicitly set. The most common case is querying the sort field value with the default sort-field ={sort} setting. Being able to vary the fallback used according to the entry type allows a more flexible approach than explicitly setting the sort field in the .bib file.

Note that if you specify a different field to use for the sort value with sort-field then the fallback for that field will be used if that field is missing. The sort fallbacks will be irrelevant if the sort field isn’t being queried. If the fallback system fails to provide a value for the field identified by sort-field then bib2gls will follow the rule given by the missing-sort-backup setting.
If you require a complex sort value that can’t be implemented by the fallback system, you can use `assign-fields` to explicitly set the `sort` field to a string expression (section 5.1). Bear in mind that if the `sort` field is actually set to a value, either in the `.bib` file or through resource options, then the `sort fallback won’t be used` and the sort fallback options describe in this section won’t have any effect.

There are other fields that `bib2gls` may want to query that won’t necessarily be set in the `.bib` file but may be inferred from another field. For example, if the `sort` field fallback references the `name` field then the `name` field will also need a fallback if it hasn’t been set.

Another possibility is that the interpreter encounters content that includes commands such as \gls. Since the interpreter can’t tell at what point in the document the first use flag is changed, \gls is treated as \glstext (and similarly \glspl is treated as \glsplural) so the `text` (or `plural`) field will be queried by the interpreter.

The commands `\newglossaryentry` and `\longnewglossaryentry` are the foundation for all commands that define glossary entries. These commands both require that either the `name` or the `parent` field are set. If the `name` is omitted, then its value is obtained from the parent entry’s name. The `description` must also be provided but may be set to empty. (Some entry types, such as `@index`, will set `description` to empty if that field is missing, but for other entry types, such as `@entry`, the `description` is required and will trigger a warning if omitted.)

All other entry definition commands, such as `\newabbreviation` and `\glsxtrnewsymbol`, internally use one of those foundation commands. In the case of `\newabbreviation` (and `\newacronym`), the `name` field is set by the style using values obtained from the `short` and/or `long` fields. This is information that `bib2gls` is unaware of and may guess incorrectly when trying to determine an appropriate value for the `name` field if it is omitted (which is typically the case) from abbreviation entry types, such as `@abbreviation` or `@acronym`.

The general `@entry` entry type, uses the same rules as `\newglossaryentry`:

- **name** If the `parent` field has been set, then the parent’s `name` field is used. If the parent’s `name` field isn’t set, then the fallback for the parent’s `name` field is used (which will depend on the parent’s entry type). If neither the `name` nor the `parent` field is set, then a warning is issued since at least one of those fields must be set for `@entry`.

- **text** If the `text` field is missing, it’s obtained from the `name` field or the fallback for the `name` field, if that hasn’t been set.

- **plural** If the `name` field has been set then the `plural` value is obtained by appending `\glspluralsuffix` to the value of the `text` field (or the fallback for the `text` field, if that hasn’t been set).

  If the `name` field hasn’t been set but the `parent` field has been set, then the `plural` is obtained from the parent’s `plural` field. If the parent’s `plural` field hasn’t been set then the fallback for that value will be used, according to the parent’s entry type.

- **first** The fallback for the `first` field is obtained from the `text` field (or the fallback for the `text` field, if that hasn’t been set).

---

2 Or the internal command that both `\newglossaryentry` and `\longnewglossaryentry` use.
The fallback for the \texttt{firstplural} field is obtained by appending \texttt{\textbackslash glsplural-suffix} to the value of the \texttt{first} field, if that field has been set, otherwise it’s obtained from the \texttt{plural} field (or the fallback for the \texttt{plural} field if that isn’t set).

Note that although \texttt{bib2gls} follows the \texttt{\textbackslash newglossaryentry} rules in order to obtain the fallback, it doesn’t explicitly set those fields in the .\texttt{glstex} file if they weren’t provided in the .\texttt{bib} file or set using options such as \texttt{replicate-fields} or \texttt{assign-fields}.

The exception to this is the \texttt{sort} field, which will be obtained from the \texttt{name} field for most entry types unless overridden by one of the applicable sort fallback options, such as \texttt{entry-sort-fallback}. If the designated fallback (such as \texttt{name}) is missing, then the fallback value for that field will be used.

The \texttt{@index} and \texttt{@indexplural} entry types are slightly different. They have their own rules for obtaining the value of the \texttt{name} field, and will explicitly set it in the .\texttt{glstex} file via the helper commands \texttt{\bibglsnewindex} and \texttt{\bibglsnewindexplural}.

In the case of \texttt{@index}, if the \texttt{name} field is missing, its value will be obtained from the entry’s original label. If the \texttt{sort} field is missing, its value is obtained from the \texttt{name} field unless a different fallback is specified with \texttt{custom-sort-fallbacks}. The remaining fallbacks are as for \texttt{@entry}.

It’s more complicated for \texttt{@indexplural}, which has the following fallback rules:

\texttt{name} If the \texttt{name} field is missing, its value is obtained from the entry’s \texttt{plural} field (or the fallback for the \texttt{plural} field, if that field is missing).

\texttt{plural} If the \texttt{plural} field is missing, its value is obtained by appending \texttt{\textbackslash glsplural-suffix} to the value of the \texttt{text} field (or the fallback for the \texttt{text} field, if that field is missing).

\texttt{text} If the \texttt{text} field is missing, its value is obtained from the entry’s original label.

\texttt{sort} If the \texttt{sort} field is missing, its value is obtained from the \texttt{name} field unless a different fallback is specified with \texttt{custom-sort-fallbacks}.

The remaining fallbacks are as for \texttt{@entry}.

The most awkward of all the entry types are, as indicated earlier, the abbreviations where the field values such as \texttt{name} and \texttt{text} are set by the abbreviation style. Therefore, there are resource options specifically to identify the most appropriate fallback values for abbreviations. The default is to use the value of the \texttt{short} field as the fallback for the \texttt{name}, \texttt{sort} and \texttt{text} fields. If this is inappropriate for your abbreviation style then you will need to use the options listed below to provide more appropriate fallbacks. These options don’t actually set the \texttt{name} and \texttt{text} fields in the .\texttt{glstex} file and don’t include any style formatting (such as font changing commands), which are irrelevant to \texttt{bib2gls}.

For other entry types, see their description in chapter 4.

\footnote{The \texttt{sort} field will be set in the .\texttt{glstex} file as it’s useful for debugging, but it’s typically irrelevant.}
5.8 Field Fallbacks

**abbreviation-name-fallback=**{field}

The entry types that define abbreviations (such as @abbreviation and @acronym) will, by default, fallback on the short field if the name field is missing and it’s required for some reason (for example, with sort-field={name}). If you prefer to fallback on a different field, then you can use this option to specify the field. For example, abbreviation-name-fallback={long}. The (field) value must be a known field (not an internal field) but can’t be the sort field.

Note that the default fallback for the sort field for abbreviations is given by abbreviation-sort-fallback which is set to short not name by default. So changing the fallback for the name field won’t have an effect unless the sort fallback is changed to name or sort-field={name} is used or the name field is referenced in an option such as assign-fields.

Field concatenation isn’t available for this option.

**abbreviation-text-fallback=**{field}

Similar to abbreviation-name-fallback but for the text field. The default fallback is the short field. Field concatenation isn’t available for this option.

Note that you can’t have both abbreviation-name-fallback={text} and abbreviation-text-fallback={name} as it would cause an infinite loop.

**abbreviation-sort-fallback=**{field}

The entry types that define abbreviations (such as @abbreviation and @acronym) will, by default, fallback on the short field if the sort field is missing (assuming sort-field={sort}). If you prefer to fallback on a different field, then you can use this option to specify the field. For example, abbreviation-sort-fallback={long}. Note that if you use sort-field={name}, then the fallback field will be given by abbreviation-name-fallback if the name field is omitted.

The (field) may be a known field but not an internal field. It can’t be the sort field. It may also be one of the keywords: id (for the entry’s label) or original id (for the entry’s original label). The (field) may also be a composite in the form (field1)+(field2)+... which indicates that the sort value should be obtained by concatenating the values of given fields, where the separator is given by field-concat-sep.

Note that missing-sort-fallback and custom-sort-fallbacks override this setting.

The abbreviation-sort-fallback setting is only used when bib2gls tries to access the sort field for an abbreviation and finds that the field hasn’t been set. This means that this setting has no effect if you explicitly set the sort field or if you change the field used for sorting (sort-field).
entry-sort-fallback=⟨field⟩

The regular entry types (such as @entry and @dualentry) will, by default, fallback on the name field if the sort field is missing (assuming sort-field={sort}). If you prefer to fallback on a different field, then you can use this option to specify the field. Note that missing-sort-fallback and custom-sort-fallbacks override this setting.

The ⟨field⟩ may be a known field but not an internal field. It can’t be the sort field. It may also be one of the keywords: id (for the entry’s label) or original id (for the entry’s original label). The ⟨field⟩ may also be a composite in the form ⟨field1⟩+⟨field2⟩+… which indicates that the sort value should be obtained by concatenating the values of given fields, where the separator is given by field-concat-sep.

This setting doesn’t affect the index type of entries, such as @index or @indexplural. This is useful if your glossary contains homographs (terms with the same spelling) which can’t be distinguished by the sort comparators. For example, suppose my file entries.bib contains:

@index{pagelist,
    name={page list},
    description={a list of individual pages or page ranges}
}

@index{glossary}

@entry{glossarylist,
    parent={glossary},
    description={list of technical words}
}

@entry{glosscol,
    parent={glossary},
    description={collection of glosses}
}

Now first consider a document that uses the default settings:

\documentclass{article}
\usepackage[record,subentrycounter,style={treenoname}]{glossaries-extra}
\GlsXtrLoadResources[src={entries}]
\begin{document}
A test document describing \glspl{pagelist} and \gls{glosscol} (collection) vs \gls{glossarylist} (list).
5.8 Field Fallbacks

\printunsrtglossary
\end{document}

The default behaviour for @entry if the sort field is missing is to fallback on the name field. If the name field is missing (as with glossarylist and glosscol), then the value is obtained from the name field from the parent entry. The parent entry for these homographs is the glossary entry, which was defined with @index and doesn’t have the name field. For the @index entries, if name is missing the value is obtained from the label.

Therefore both glossarylist and glosscol end up with the same sort value: glossary. This triggers a message in verbose mode (--verbose) which can be found in the transcript file:

Identical sort values for 'glossarylist' and 'glosscol'
Falling back on ID

So the actual sort values used are "glossarylist" and “glosscol”. This puts the glossarylist entry before the glosscol entry.

Now suppose a minor modification is made to the document:

\GlsXtrLoadResources
[  
  src={entries},
  entry-sort-fallback={description}
]

This means that when the sort function fails to find the sort field for the terms defined with @entry, it will fallback on the description field. This doesn’t affect the terms defined with @index, which still fallback on the name field. This time there’s no message in the transcript file and the glosscol entry now comes before the glossarylist entry.

The entry-sort-fallback setting is only used when bib2gls tries to access the sort field for a term defined with @entry and finds that the field hasn’t been set. This means that this setting has no effect if you explicitly set the sort field or if you change the field used for sorting (sort-field).

symbol-sort-fallback=⟨field⟩

The entry types that define symbols (such as @symbol and @number) will, by default, fallback on the entry’s original label if the sort field is missing (assuming the default sort-field= {sort}). If you prefer to fallback on a different field, then you can use this option to specify the field. For example, symbol-sort-fallback={name}.

The ⟨field⟩ may be a known field but not an internal field. It can’t be the sort field. It may also be one of the keywords: id (for the entry’s label) or original id (for the entry’s original label). The ⟨field⟩ may also be a composite in the form ⟨field1⟩+⟨field2⟩+… which
indicates that the sort value should be obtained by concatenating the values of given fields, where the separator is given by field-concat-sep.

Note that missing-sort-fallback and custom-sort-fallbacks override this setting.

The symbol-sort-fallback setting is only used when bib2gls tries to access the sort field for a symbol and finds that the field hasn’t been set. This means that this setting has no effect if you explicitly set the sort field or if you change the field used for sorting (sort-field).

bibtexentry-sort-fallback=⟨field⟩

The main @bibtexentry entry types will, by default, fallback on the name if the sort field is missing (assuming the default sort-field={sort}). If you prefer to fallback on a different field, then you can use this option to specify the field.

The ⟨field⟩ may be a known field but not an internal field. It can’t be the sort field. It may also be one of the keywords: id (for the entry’s label) or original id (for the entry’s original label). The ⟨field⟩ may also be a composite in the form ⟨field1⟩+⟨field2⟩+… which indicates that the sort value should be obtained by concatenating the values of given fields, where the separator is given by field-concat-sep.

Note that missing-sort-fallback and custom-sort-fallbacks override this setting.

The bibtexentry-sort-fallback setting is only used when bib2gls tries to access the sort field for a main entry defined with @bibtexentry and finds that the field hasn’t been set. This means that this setting has no effect if you explicitly set the sort field or if you change the field used for sorting (sort-field).

custom-sort-fallbacks={⟨key=value list⟩}

The value should be a key=value list in the form ⟨entrytype⟩=⟨field⟩ where ⟨entrytype⟩ is the original entry type (before being aliased with entry-type-aliases). This will override any of the sort fallback options listed below for entries whose original entry type matches ⟨entrytype⟩.

The ⟨field⟩ may be a known field but not an internal field. For obvious reasons, it can’t be the sort field (since ⟨field⟩ is the fallback a missing sort field). It may also be one of the keywords: id (for the entry’s label) or original id (for the entry’s original label). The ⟨field⟩ may also be a composite in the form ⟨field1⟩+⟨field2⟩+… which indicates that the sort value should be obtained by concatenating the values of the given fields, where the separator is given by field-concat-sep.

For example, if the .bib file contains:

@unit{ohm,
  name={\si{\ohm}},
  description={electrical resistance}}
5.8 Field Fallbacks

```latex
}\constant{\pi,
   \name{\pi},
   \description{the ratio of the length of the circumference of a circle to its diameter},
   \user1{3.14159}
}\}

\symbol{fx,
   \name{f(x)},
   \description{a function of $x$}
}\}

\number{zero,
   \name{0},
   \description{nothing or no quantity}
}\}

Then the resource options:

\entry-type-aliases={\unit=symbol,\constant=number},
\custom-sort-fallbacks={\unit=name,\constant=user1}

will treat the custom @unit and @constant entries as though they had been defined with @symbol and @number, respectively, but the fallback for the sort field is different: the ohm entry will use the name field for the sort fallback (because its original entry type was unit), the pi entry will use the user1 field for the sort fallback (because its original entry type was constant) and the fx and zero entries will use the label for the sort fallback (since neither symbol nor number were identified in custom-sort-fallbacks so the symbol-sort-fallback is used).

If an entry hasn’t had its entry type aliased then (entrytype) is its actual entry type. For example, consider the following definitions:

@abbreviation{svm,
   \short={SVM},
   \long={support vector machine}
}\}

@acronym{laser,
   \short={laser},
   \long={light amplification by stimulated emission of radiation}
}\}

Then abbreviation-sort-fallback={short} will make both entries fallback on the short field (since abbreviation-sort-fallback applies to both @acronym and @abbreviation), but the option:
5.8 Field Fallbacks

custom-sort-fallbacks={abbreviation=long,acronym=short}

will make the entry defined with @abbreviation fallback on the long field and the entry defined with @acronym will fallback on the short field.

Since the default setting is abbreviation-sort-fallback={short} this only needs to be:

custom-sort-fallbacks={abbreviation=long}

In this case, the entry defined with @abbreviation ("SVM") will use the setting given in custom-sort-fallbacks, but the entry defined with @acronym ("laser") will use the setting given by abbreviation-sort-fallback since @acronym hasn’t been identified in custom-sort-fallbacks.

This option also covers dual entries. For example:

custom-sort-fallbacks={
    dualindexnumber=description,
    dualindexnumbersecondary=user1
}

Note that the entry type for the dual is in the form ⟨primary entry type⟩secondary.

The custom-sort-fallbacks setting is only used when bib2gls tries to access the sort field for an entry (whose original entry type has been identified in this setting) and finds that the field hasn’t been set. This means that this setting has no effect if you explicitly set the sort field or if you change the field used for sorting (sort-field).

field-concat-sep=⟨value⟩

This option sets the field concatenation separator to ⟨value⟩ used by the sort fallback options. The default is a space. An empty value indicates no separator. You may use \u⟨hex⟩ to indicate a character by its hexadecimal code (see section 1.5). Note that the more complex field concatenation specification described in section 5.1 isn’t available for this option.

For example, suppose the .bib file contains:

@abbreviation{ac,
    short={AC},
    long={alternating current}
}
@index{acacia}

Then the resource option:

\GlsXtrLoadResources[
    sort={letter-nocase},
    abbreviation-sort-fallback={short+long}
]

5.9 Plurals

will set the ac sort value to “AC alternating current”. That is, the short value concatenated with the long value using the default space separator. With the letter-nocase sort method, this will put the ac entry before the acacia entry (because the space character comes before “a”).

If the resource options are changed to:

```
\GlsXtrLoadResources[
  sort={letter-nocase},
  field-concat-sep={},
  abbreviation-sort-fallback={short+long}
]
```

This will obtain the sort value for abbreviations from a concatenation of the ⟨short⟩ and ⟨long⟩ values without a separator. This means that the ac sort value will be “AC alternating current” and so the ac entry will come after the acacia entry (since “I” comes after “c”).

This setting is only used for the sort fallback options that allow field concatenation (such as entry-sort-fallback but not missing-sort-fallback).

Note that due to the way that the key=value list parser trims leading and trailing spaces, you can’t simply do `field-concat-sep={} ` to indicate a space character as the value will end up as an empty string. You can instead do `field-concat-sep={\string\u20}` but since this is the default value there shouldn’t be much need for it.

Remember that the separator may be replaced with a break point marker depending on the sort method and break-at setting.

5.9 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, mice), so a simplistic approach of just doing `\gls{⟨label⟩}[s]` will sometimes produce inappropriate results, so the glossaries package provides a plural key with the corresponding command `\glspl`.

In some cases a plural may not make any sense (for example, if the term is a verb or symbol), so the plural key is optional, but to make life easier for languages where the majority of plurals can simply be formed by appending a suffix to the singular, the glossaries package lets the plural field default to the value of the text field with `\glspluralsuffix` appended. This command is defined to be just the letter “s”. This means that the majority of terms in such languages don’t need to have the plural supplied as well, and you only need to use it for the exceptions.

For languages that don’t have this general rule, the plural field will always need to be supplied for nouns.

There are other plural fields, such as firstplural, longplural and 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:
5.9 Plurals

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

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

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

The last case is different with the glossaries-extra extension package. The `shortplural` field defaults to the `short` field with \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, \abbrvpluralsuffix is redefined. Most styles simply define this command as:

\renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}

where \glsxtrabbrvpluralsuffix expands to \glspluralsuffix. The “sc” styles (such as long-short-sc) use a different definition:

\renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}

This allows the suffix to be reverted back to the upright font, counteracting the affect of the small-caps font.

This means that if you want to change or strip the suffix used for the plural short form, it’s usually not sufficient to redefine \abbrvpluralsuffix, as the change will be undone the next time the style is applied. Instead, for a document-wide solution, you need to redefine \glsxtrabbrvpluralsuffix. Alternatively you can use the category attributes.

There are two attributes that affect the short plural suffix formation. The first is `aposplural` which uses the suffix `\abbrvpluralsuffix`. That is, an apostrophe followed by \abbrvpluralsuffix is appended. The second attribute is `noshortplural` which suppresses the suffix and simply sets `shortplural` to the same as `short`.

With bib2gls, if you have some abbreviations where the plural should have a suffix and some where the plural shouldn’t have a suffix (for example, the document has both English and French abbreviations) then there are two approaches.

The first approach is to use the category attributes. For example:

\glssetcategoryattribute{french}{noshortplural}

Now just make sure all the French abbreviations are have their `category` field set to `french`:

226
The other approach is to use the options listed below for the given resource set. For example:

\GlsXtrLoadResources[src={fr-abbrvs},short-plural-suffix={}]  

**short-plural-suffix={value}**

Sets the plural suffix for the default `shortplural` to `{value}`. The `{value}` may be one of:

- `{suffix}`: add the `shortplural` field, if missing, with the given `{suffix}`.
- `{empty}`: add the `shortplural` field, if missing, with no suffix.
- `use-default`: leave it to `glossaries-extra` to determine the appropriate default.

The default setting is `short-plural-suffix={use-default}`. If the `={value}` part is omitted, then `short-plural-suffix={} is assumed.`

**dual-short-plural-suffix={value}**

Sets the plural suffix for the default `dualshortplural` field to `{value}`. As with `short-plural-suffix`, the default setting is `dual-short-plural-suffix={use-default}`. If the `{value}` is omitted or empty, the suffix is set to empty.

---

### 5.10 Location List Options

The **record** package option automatically adds two new keys: `loclist` and `location`. These two fields are set by `bib2gls` from the information supplied in the `.aux` file (unless the option `save-locations={false}` is used). The `location` field contains the code to typeset the formatted location list.

Note that the cross-referencing information provided with the `see`, `seealso` and `alias` fields is put in the location list. If you only want the cross-reference and not any of the locations, use `save-locations={see}` (or similar).

The `loclist` field has the syntax of an `etoolbox` internal list and includes every location (except for the discarded duplicates and ignored records) with no range formations. Any explicit range markup is stripped from the `format` information to leave just the `ENCAP` name, so you just get the start and end locations added as individual elements but they are still encapsulated with the associated formatting command. Each item in the list is provided in one of the following forms:

```
\glsseeformat[{tag}]{⟨label list⟩}{}
```

for the cross-reference supplied by the `see` field,

```
\glsxtruseseealsoformat{⟨xr list⟩}
```

for the cross-reference supplied by the \seealso field,
\glsnoidxdisplayloc{{\langle prefix\rangle}\{\langle counter\rangle\}{\langle format\rangle}\{\langle location\rangle\}}
for standard the internal locations,
\glsxtrdisplaysupplloc{{\langle prefix\rangle}\{\langle counter\rangle\}{\langle format\rangle}\{\langle src\rangle\}{\langle location\rangle}}
for supplemental (external) locations and
\glsxtrdisplaylocnameref{{\langle prefix\rangle}\{\langle counter\rangle\}{\langle format\rangle}\{\langle location\rangle\}\{\langle title\rangle\}}\{\langle href\rangle\}\{\langle hcounter\rangle\}\{\langle file\rangle\}
for nameref records. (See section 5.11 for more information about supplemental locations and --merge-nameref-on for more information about nameref records.) You can iterate through the \loclist value using one of etoolbox's internal list loops (either by first fetching the list using \glsfieldfetch or through glossaries-extra's \glsxtrfield-dolistloop or \glsxtrfieldforlistloop shortcuts).

The \langle format\rangle is that supplied by the \format key when using commands like \gls or \glsadd (the encapsulator or ENCAP in makeindex parlance). If omitted, the default \format =\{\glsnumberformat\} is assumed (unless this default value is changed with \GlsXtrSetDefaultNumberFormat. The value of the \format key must be the name of a text-block command without the leading backslash that takes a single argument (the location). The location is encapsulated by that command. For example,
\gls[format={textbf}]\{sample\}
will display the corresponding location in bold, but note that this will no longer have a hyperlink if you've used \hyperref. If you want to retain the hyperlink you need the location encapsulated with \hyperbf instead of \textbf:
\gls[format={hyperbf}]\{sample\}
The \hyper\langle xx\rangle set of commands all internally use \glshypernumber which adds the appropriate hyperlink to the location. See Table 6.1 in the glossaries [14] user manual for a list of all the \hyper\langle xx\rangle commands.
Ranges can be explicitly formed using the parenthetical syntax \format=\{()\} and \format=\{() or format=\{\langle csname\rangle\} and format=\{()\langle csname\rangle\} (where \langle csname\rangle is again the name of a text-block command without the initial backslash) in the optional argument of commands like \gls or \glsadd. With glossaries-extra v1.50+, you can also use \glsstart-range and \glsend-range (which is useful if the unbalanced parentheses upset syntax highlighting).

These explicit ranges will always form a range, regardless of \min-loc-range, unless the start and end coincide and --collapse-same-location-range is in effect. The explicit range will be encapsulated with \bibglsrange (unless merge-ranges={true}). (This command is not used with implicit ranges that are formed by collating consecutive locations.)
5.10 Location List Options

The initial marker is stripped from the \texttt{⟨format⟩} argument of the location formatting commands, such as \glsnoidxdisplayloc, to allow for easy conversion to the corresponding text-block command.

Explicit ranges don’t merge with neighbouring locations (unless \texttt{merge-ranges={true}}), but will absorb any individual locations within the range that doesn’t conflict. (Conflicts, denoted interlopers, will be moved to the start of the explicit range, regardless of \texttt{merge-ranges}.) For example, if \gls{sample} is used on page 1, \gls{sample} is used on page 2, \gls{sample} is used on page 3, and \gls{sample} is used on page 4, then the location list will be 1, 2–4. The entry on page 3 is absorbed into the explicit range, but, with the default \texttt{merge-ranges={false}}, the range can’t be expanded to include page 1. If the entry on page 3 had a different format to the explicit range, for example \gls{sample} then this will cause a warning and the interloper will be moved before the start of the range so that the location list would then be 1, 3, 2–4.

The \texttt{merge-ranges={true}} option will make explicit ranges behave like implicit ranges, which allows them to merge with neighbouring ranges. The \texttt{\bibglsrange} command won’t be used in this case (regardless of whether or not the range was merged with neighbouring locations). Options such as \texttt{min-loc-range} won’t have an effect on the merged range, but will still effect implicit ranges that haven’t been merged with an explicit range.

An ignored record identifies a term that needs to be treated as though it has a record for selection purposes, but the record should not be included in the location list. The special format \texttt{format={glsignore}} is provided by the glossaries package for cases where the location should be ignored. (The command \texttt{glsignore} simply ignores its argument.) This works reasonably well if an entry only has the one location, but if the entry happens to be indexed again, it can lead to an odd empty gap in the location list with a spurious comma. If \texttt{bib2gls} encounters a record with this special format, the entry will be selected but the record will be discarded.

This means that the location list will be empty if the entry was only indexed with the special ignored format, but if the entry was also indexed with another format then the location list won’t include the ignored records. (This format is used by \texttt{\glsaddallunused} but remember that iterative commands like this don’t work with \texttt{bib2gls}. Instead, just use \texttt{selection={all}} to select all entries. Those that don’t have records won’t have a location list.)

For example, suppose you only want main matter locations in the number list, but you want entries that only appear in the back matter to still appear in the glossary (without a location list), then you could do:

\begin{verbatim}
\backmatter
\GlsXtrSetDefaultNumberFormat{glsignore}
\end{verbatim}

If you also want to drop front matter locations as well:

\begin{verbatim}
\frontmatter
\GlsXtrSetDefaultNumberFormat{glsignore}
\end{verbatim}

\begin{verbatim}
...
\mainmatter
\end{verbatim}
5.10 Location List Options

\GlsXtrSetDefaultNumberFormat{glsnumberformat}
...
\backmatter
\GlsXtrSetDefaultNumberFormat{glsignore}

Note that explicit range formations aren’t discarded, so if \glsignore is used in a range, such as:
\glsadd[format={\glsignore}]{sample}
...
\glsadd[format={}\glsignore]{sample}

then the range will be included in the location list (encapsulated with \glsignore), but this case would be a rather odd use of this special format and is not recommended.

The record counting commands, such as \rgls, use the special format glstriggerrecord-format, which bib2gls also treats as an ignored record and the same rules as for glsignore apply.

The locations are always listed in the order in which they were indexed, (except for the cross-reference which may be placed at the start or end of the list or omitted). This is different to xindy and makeindex where you can specify the ordering (such as lower case Roman first, then digits, etc), but unlike those applications, bib2gls allows any location, although it may not be able to work out an integer representation. (With xindy, you can define new location formats, but you need to remember to add the appropriate code to the custom module.)

It’s possible to define a custom glossary style where \glossentry (and the child form \subglossentry) ignore the final argument (which will be the location field) and instead parse the loclist field and re-order the locations or process them in some other way. Remember that you can also use \glsnoidxloclist provided by glossaries. For example:
\glsfieldfetch{gls.sample}{loclist}{\loclist}% fetch location list
\glsnoidxloclist{\loclist}% iterate over locations

This uses \glsnoidxloclisthandler as the list’s handler macro, which simply displays each location separated by \delimN. (See also Iteration Tips and Tricks [16].)

Each regular location is listed in the .aux file in the form:
\glsxtr@record{⟨label⟩}{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}

(See --merge-nameref-on for nameref records.) Exact duplicates are discarded. For example, if cat is indexed twice on page 1:
\glsxtr@record{cat}{}{page}{glsnumberformat}{1}
\glsxtr@record{cat}{}{page}{glsnumberformat}{1}

then the second record is discarded. Only the first record is added to the location list.

Partial duplicates, where all arguments match except for ⟨format⟩, may be discarded depending on the value of ⟨format⟩. For example, if page 1 of the document uses \gls{cat} and \gls[format=\hyperbf]{cat} then the .aux file will contain:
This is a partial record match. In this case, bib2gls makes the following tests:

- If one of the formats includes an explicit range formation marker, the range takes precedence.

- If one of the formats is `glsnumberformat` (as in the above example) or an ignored record format such as `glsignore`, that format will be skipped. So in the above example, the second record will be added to the location list, but not the first. (A message will only be written to the transcript if the `--debug` switch is used.) The default `glsnumberformat` will take precedence over the ignored record formats (`glsignore` and `glstriggerrecordformat`).

- If a mapping has been set with the `--map-format` switch that mapping will be checked.

- Otherwise the duplicate record will be discarded with a warning.

The location field is used to store the formatted location list. The code for this list is generated by bib2gls based on the information provided in the .aux file, the presence of the `see` or `seealso` field and the various settings described in this chapter. When you display the glossary using \printunsrtglossary, if the location field is present it will be displayed according to the glossary style (and other factors, such as whether the `nonumberlist` option has been used, either as a package option or supplied in the optional argument of \printunsrtglossary). For more information on adjusting the formatting see the glossaries [14] and glossaries-extra [13] user manuals.

\save-locations={value}

This was originally a boolean setting, but as from v3.0 there are additional values.

- **false**: don’t save anything in the location field;
- **true**: save cross-references and all non-ignored locations in the location field;
- **see**: only save cross-references (see, seealso and alias) in the location field;
- **see not also**: only save the see and alias cross-references (not seealso) in the location field;
- **alias only**: only save the alias cross-references (not see or seealso) in the location field.

By default, the locations will be processed and stored in the location and loclist fields. However, if you don’t want the location lists (for example, you are using the `nonumberlist` option or you are using xindy with a custom location rule), then there’s no need for bib2gls to process the locations. To switch this function off, just use `save-locations={false}`.
Note that with this setting, if you’re not additionally using \makeindex or \xindy, then the locations won’t be available even if you don’t have the \nonumberlist option set.

The boolean \nonumberlist key that may be used in \newglossaryentry can also be used in a .bib file, but in this case it can’t have an empty value. The value must be either \texttt{true} or \texttt{false}. If true then \texttt{bib2gls} won’t save the \texttt{location} or \texttt{loclist} fields, regardless of the \save-locations resource option.

The \nonumberlist key provided by the base glossaries package doesn’t represent a real field. The value isn’t saved but, if used, it will alter the indexing information that’s written to the \makeindex or \xindy file. It’s a little hack to ensure that the location is hidden for a specific entry when used with \makeindex and \xindy.

\texttt{bib2gls} will look for this key to determine if the location should be omitted for the given entry, but it won’t write the key to the \glstex file.

\texttt{save-loclist=\langle boolean\rangle}

If you want the \texttt{location} field but don’t need \texttt{loclist}, you can use \texttt{save-loclist=\{false\}}. This can help to save resources and build time.

\texttt{save-primary-locations=\langle value\rangle}

A synonym for \texttt{save-principal-locations}.

\texttt{save-principal-locations=\langle value\rangle}

It’s sometimes useful to identify a principal location with a different format, such as bold or italic. This helps the reader select which location to try first in the event of a long location list. However, you may prefer to store the principal locations in a different field to give it a more prominent position. In order to do this you need to specify the format (or formats) used to identify principal locations with \texttt{principal-location-formats} and use \texttt{save-principal-locations} to determine how to deal with these locations.

This option may take one of the following values:

- \texttt{false}: don’t save principal locations (default);
- \texttt{retain}: save principal locations in the \texttt{primarylocations} field but don’t remove from the usual location list;
- \texttt{default format}: similar to \texttt{retain} but the format for the principal records in the \texttt{location} field is converted to the default \texttt{glsnumberformat ENCAP} (the records in the \texttt{primarylocations} field retain their given format);
- \texttt{start}: save principal locations in the \texttt{primarylocations} field and also move to the start of the usual location list;
5.10 Location List Options

- remove: save principal locations in the `primarylocations` field and remove from the usual location list. You may want to consider using the –retain-formats switch with this setting if you don’t want to lose a partial location match (for example, if the principal location coincides with the start of an explicit range).

The principal locations are copied to the `primarylocations` field and are either encapsulated with `{bibglprimary}` or can be split into groups, according to `principal-loc-counters`.

If you use `save-principal-locations={remove}`, the `location` field will end up empty if the locations for the associated entry were all identified as principal. If you use `save-principal-locations={start}`, all principal locations will be moved to the start of the location list stored in the `location` field, but there will be no additional markup (other than the given format) to identify them. If you need additional markup, then use `save-principal-locations={remove}` and adjust the location list format to insert the principal locations at the start. This can be done by modifying the glossary style.

For example, the bookindex style inserts `{glsxtrbookindexprelocation}` before the location, so you could redefine this:

```latex
\renewcommand*{\glsxtrbookindexprelocation}[1]{%
  \glsxtrifhasfield{primarylocations}{#1}{% 
    \glsxtrprelocation
    \globalscurrentfieldvalue
    \glsxtrifhasfield{location}{#1}{;}{;}{% 
    }%
  }%
  \glsxtrprelocation
}
```

(Note that if `loc-prefix` is used, the prefix will be in the `location` field and so will come after the principal locations in the above example. Similarly for cross-references unless they’ve been omitted.)

You can switch from using the `location` field to the `primarylocations` field by locally changing \GlsXtrLocationField:

```latex
\printunsrtglossary*{%
  \renewcommand{\GlsXtrLocationField}{primarylocations}%
}
```

Remember that the handler used by \printunsrtglossary will fallback on the `loclist` field if the field identified by \GlsXtrLocationField is missing or empty. You may want to consider using `save-loclist={false}` to prevent this.

`primary-location-formats={list}`

A synonym for `principal-location-formats`. 233
5.10 Location List Options

principal-location-formats=⟨list⟩

This option will automatically set save-principal-locations={retain} unless it has already been changed from the default save-principal-locations={false} setting. The argument should be a comma-separated list of formats. If a record’s format is contained in this list then it will be considered a principal location and it will be included in the associated entry’s primarylocations field.

For example, suppose the file entries.bib contains:

@entry{bird,  
  name={bird},  
  description={feathered animal}  
}

@entry{waterfowl,  
  name={waterfowl},  
  description={any bird that lives in or about water}  
}

@entry{zebra,  
  name={zebra},  
  description={striped African horse}  
}

@entry{parrot,  
  name={parrot},  
  description={mainly tropical bird with bright plumage}  
}

and the document test.tex contains:

\documentclass{report}

\usepackage[colorlinks]{hyperref}
\usepackage[record,  
  postpunc={dot},  
  nostyles,  
  stylemods={tree,bookindex},  
  style={bookindex}]{glossaries-extra}

\GlsXtrLoadResources[  
  src={entries},  
  principal-location-formats={hyperbf,hyperemph},  
  save-principal-locations={remove}  
]

\renewcommand*{\glsxtrbookindexprelocation}[1]{%  
  \glsxtrifhasfield{primarylocations}{#1}%
}
5.10 Location List Options

The principal-location-formats={hyperbf, hyperemph} setting in the above indicates that locations encapsulated with \hyperbf and \hyperemph are principal records. In this case, the bold format is used to indicate the principal location in the main document text and the emphasized format is used to indicate the location in the main glossary.

The principal records are removed from the location field due to the save-principal-locations={remove} setting. This can lead to a ragged location list. The option save-principal-locations={default format} can allow the principal location to be absorbed into a range.

The main glossary records are added through the category-independent post-name hook with \glsadd. This won’t be implemented until the entries are actually defined as the page number can’t be determined until the glossary can be displayed. This means that the docu-
ment build requires an extra `bib2gls` and \texttt{pdflatex} run:

```
pdflatex test
bib2gls --group test
pdflatex test
bib2gls --group test
pdflatex test
```

For consistency, I’ve used \texttt{\glsxtrnewglslike} to provide commands used to indicate a principal reference in the text. This means that if I decide to change the optional arguments used for principal references I only need to edit one line. For example, I might want to change the default counter:

\texttt{\glsxtrnewglslike[format={hyperbf},counter={chapter}]{\primary}\{\primarypl}\{\Primary}\{\Primarypl}\}

Here’s another example that only has one principal format (\texttt{hyperrm}) that’s indexed through the use of \texttt{\GlsXtrAutoAddOnFormat}, which sets up a hook that automatically inserts:

\texttt{\glsadd[counter={chapter},format={hyperrm}]{\langle\langle label\rangle\rangle}}

on each instance of \texttt{\gls[format={primaryfmt}]{\langle\langle label\rangle\rangle}} (or similar). This means that the entry is indexed twice when this particular format is used: first with the \texttt{hyperrm} format and chapter counter (from the \texttt{\glsadd} command in the hook), and then with the \texttt{primaryfmt} format and the default counter (as per normal behaviour):

```
\documentclass{report}

\usepackage[colorlinks]{hyperref}
\usepackage[
  record={nameref},
  postpunc={dot},
  nostyles,
  stylemods={tree,bookindex},
  style={bookindex}]{glossaries-extra}

\GlsXtrLoadResources[
  src={topics},
  principal-location-formats={hyperrm},
  save-principal-locations={remove},
  save-loclist={false}
  ]

\newcommand{\primaryfmt}[1]{\hyperbf{#1}}

\GlsXtrAutoAddOnFormat{primaryfmt}{counter={chapter},format={hyperrm}}
```
5.10 Location List Options

\begin{document}
\chapter{Sample}
\primary{waterfowl}, \gls{bird} and \gls{zebra}.

\chapter{Another Sample}
\Gls{waterfowl}, \primary{bird} and \gls{zebra}.

\chapter{Yet Another Sample}
\Gls{waterfowl}, \gls{bird} and \primary{zebra}.

\chapter{Yet Another Sample Again}
\Gls{waterfowl}, \gls{bird}, \primarypl{parrot} and \gls{zebra}.

\printunsrtglossary*[style={tree},title={Summary}]{%\renewcommand*{\glsextrapostnamehook}{\glsadd[format={hyperemph}]{#1}}%\renewcommand{\GlsXtrLocationField}{primarylocations}}%
\printunsrtglossary[title={Index},target={false}]%\end{document}

Note that in this case, from \texttt{bib2gls}' point of view, the principal format is \texttt{hyperrm} not \texttt{primaryfmt}. This picks out the records created with the automated \texttt{\glsadd}, which have the counter set to chapter. The first glossary (with the title “Summary”) switches the location field to \texttt{primarylocations} so that only the principal records are listed. Since \texttt{record={nameref}} has been used this means that the chapter title is shown rather than the chapter number.

The second glossary (“Index”) shows the location lists that only have the page counter (because the automated \texttt{\glsadd} records with the chapter counter have been removed because they were identified as principal records). These just show the page number as that’s the default display with \texttt{record={nameref}} for records with the page counter.

An alternative to \texttt{\GlsXtrAutoAddOnFormat} would be to simply define the custom commands as follows:

\begin{verbatim}
\newcommand{\primary}[2][]\{%
\glsadd[counter={chapter},format={hyperrm}]{#2}%
\gls[format={primaryfmt},#1]{#2}%
\}
\newcommand{\primarypl}[2][]\{%
\glsadd[counter={chapter},format={hyperrm}]{#2}%
\}
\end{verbatim}
This is more useful if you want to simply omit the \texttt{format=primaryfmt} option (just remove it from the above four definitions), which makes it easier to merge the locations into ranges in the index.

\texttt{primary-loc-counters=}\langle\texttt{value}\rangle

A synonym for \texttt{principal-loc-counters}.

\texttt{principal-loc-counters=}\langle\texttt{value}\rangle

This option determines whether the principal locations should be split into groups according to the location counter. The value may be one or:

- combine: don’t split into groups (default);
- match: match the \texttt{loc-counters} setting;
- split: split into groups regardless of the \texttt{loc-counters} setting.

With \texttt{principal-loc-counters=}\langle\texttt{combine}\rangle or with \texttt{principal-loc-counters=}\langle\texttt{match}\rangle and the default \texttt{loc-counters=}\langle\texttt{as use}\rangle settings, no groups will be formed and the principal locations will be encapsulated with \texttt{\bibglsprimary}. Otherwise, the locations will be split into groups according to the counter and each group will be encapsulated with \texttt{\bibglsprimarylocationgroup} and separated with \texttt{\bibglsprimarylocationgroupsep}.

For example, suppose the file \texttt{topics.bib} contains the following entry:

\begin{verbatim}
@entry{zebra,
  name={zebra},
  description={striped African horse}
}
\end{verbatim}

The document sets up a principal location format identified by the custom command \texttt{\primaryfmt}:

\begin{verbatim}
\newcommand{\primaryfmt}[1]{\hyperbf{#1}}
\end{verbatim}
The \GlsXtrAutoAddOnFormat command is used to automatically record an entry with the chapter counter (using \glsadd) every time the entry is recorded with the principal location format:

\GlsXtrAutoAddOnFormat{primaryfmt}{counter=chapter, format=primaryfmt}

This means that, for example,

\gls[format=primaryfmt]{⟨zebra⟩}

will also first do:

\glsadd[counter=chapter, format=primaryfmt]{⟨zebra⟩}

If resource set is loaded with:

\GlsXtrLoadResources[src={topics},
primary-location-formats={primaryfmt},
save-primary-locations={retain}]

then both the location field and the primarylocations field will include both the page and chapter records mixed together. The primarylocations field will have the locations encapsulated with \bibglspprimary.

With

\GlsXtrLoadResources[src={topics},
primary-location-formats={primaryfmt},
save-primary-locations={retain},
primary-loc-counters={split}]

the primarylocations field will have the locations split into two groups, each encapsulated with \bibglspprimarylocationgroup. The location field will have the chapter and page locations intermingled.

With

\GlsXtrLoadResources[src={topics},
primary-location-formats={primaryfmt},
save-primary-locations={retain},
primary-loc-counters={split},
loc-counters={page}]

The primarylocations field will be the same as before, but the location field will only have the page locations.

Whereas with
Both the primarylocations field and the location field will only have the page locations. The order of the groups depends on whether split or match is used. With primary-loc-counters={match} the counter group order will match loc-counters. Whereas with primary-loc-counters={split} the counter group order will be determined by the order of records.

So in the case of the above document where the chapter record is automatically added before the page record (where the format is primaryfmt) then with:

\GlsXtrLoadResources[src={topics},
primary-location-formats={primaryfmt},
save-primary-locations={retain},
primary-loc-counters={match},
loc-counters={page,chapter}
]

then both the location field and the primarylocations field will have the page group first, followed by the chapter group. Whereas with:

\GlsXtrLoadResources[src={topics},
primary-location-formats={primaryfmt},
save-primary-locations={retain},
primary-loc-counters={skip},
loc-counters={page,chapter}
]

then the primarylocations field will have the chapter group first, followed by the page group.

merge-ranges={boolean}

This boolean option determines whether or not explicit ranges should merge with neighbouring locations on either side of the range. The default setting is merge-ranges={false}.

Note that \bibglsrange won’t be used with merge-ranges={true}, regardless of whether or not the range was merged with neighbouring locations. Options such as min-loc-range, suffixF and suffixFF won’t have an effect on the merged range, but will still effect implicit ranges that haven’t been merged with an explicit range.

Regardless of the value of this option, interlopers will still be moved to the start of the range and encapsulated with \bibglsinterloper.
min-loc-range={value}

By default, three or more consecutive locations ⟨loc-1⟩, ⟨loc-2⟩, ..., ⟨loc-n⟩ are compressed into the range ⟨loc-1⟩ delimR ⟨loc-n⟩ (an implicit range). Otherwise the locations are separated by \bibglsdelimN or \bibglslastDelimN. As mentioned above, these aren't merged with explicit range formations unless merge-ranges={true}.

You can change how many consecutive locations are need to form an implicit range with the min-loc-range setting where ⟨value⟩ is either none (don’t form ranges) or an integer greater than one indicating how many consecutive locations should be converted into a range.

\bib2gls determines if one location {⟨prefix-2⟩}{⟨counter-2⟩}{⟨format-2⟩}{⟨location-2⟩} is one unit more than another location {⟨prefix-1⟩}{⟨counter-1⟩}{⟨format-1⟩}{⟨location-1⟩} according to the following:

1. If ⟨prefix-1⟩ is not equal to ⟨prefix-2⟩ or ⟨counter-1⟩ is not equal to ⟨counter-2⟩ or ⟨format-1⟩ is not equal to ⟨format-2⟩, then the locations aren’t considered consecutive.

2. If either ⟨location-1⟩ or ⟨location-2⟩ are empty, then the locations aren’t considered consecutive.

3. If both ⟨location-1⟩ and ⟨location-2⟩ match the pattern (line break for clarity only)\footnote{The Java class \p{javaDigit} used in the regular expression will match any digits in the Unicode “Number, Decimal Digit” category not just the digits in the Basic Latin set. Similarly \p{javaAlphabetic} will also match alphabetic characters outside the Basic Latin set.}

   \[(.\*)?(?:\protect\s*)?\(\(\p{javaDigit}\p{javaAlphabetic}\p{javaDigit}\p{javaAlphabetic}\)+)\]

   then:

   • if the control sequence matched by group 2 isn’t the same for both locations, the locations aren’t considered consecutive;

   • if the argument of the control sequence (group 3) is the same for both locations, then the test is retried with ⟨location-1⟩ set to group 1 of the first pattern match and ⟨location-2⟩ set to group 1 of the second pattern match;

   • otherwise the test is retried with ⟨location-1⟩ set to group 3 of the first pattern match and ⟨location-2⟩ set to group 3 of the second pattern match.

4. If both ⟨location-1⟩ and ⟨location-2⟩ match the pattern

   \[(.\*)\(\p{javaDigit}\p{javaDigit}\)+\]

   then:

   a) if group 3 of both pattern matches are equal then:
5.10 Location List Options

i. if group 3 isn’t zero, the locations aren’t considered consecutive;

ii. if the separators (group 2) are different the test is retried with \( \langle \text{location-1} \rangle \) set to the concatenation of the first two groups \( \langle \text{group-1} \rangle \langle \text{group-2} \rangle \) of the first pattern match and \( \langle \text{location-2} \rangle \) set to the concatenation of the first two groups \( \langle \text{group-1} \rangle \langle \text{group-2} \rangle \) of the second pattern match;

iii. if the separators (group 2) are the same the test is retried with \( \langle \text{location-1} \rangle \) set to the first group \( \langle \text{group-1} \rangle \) of the first pattern match and \( \langle \text{location-2} \rangle \) set to the first group \( \langle \text{group-1} \rangle \) of the second pattern match.

b) If \( \langle \text{group-1} \rangle \) of the first pattern match (of \( \langle \text{location-1} \rangle \)) doesn’t equal \( \langle \text{group-1} \rangle \) of the second pattern match (of \( \langle \text{location-2} \rangle \)) or \( \langle \text{group-2} \rangle \) of the first pattern match (of \( \langle \text{location-1} \rangle \)) doesn’t equal \( \langle \text{group-2} \rangle \) of the second pattern match (of \( \langle \text{location-2} \rangle \)) then the locations aren’t considered consecutive;

c) If \( 0 < l_2 - l_1 \leq d \) where \( l_2 \) is \( \langle \text{group 3} \rangle \) of the second pattern match, \( l_1 \) is \( \langle \text{group 3} \rangle \) of the first pattern match and \( d \) is the value of \( \text{max-loc-diff} \) then the locations are consecutive otherwise they’re not consecutive.

5. The next pattern matches for \( \langle \text{prefix} \rangle \langle \text{sep} \rangle \langle n \rangle \) where \( \langle n \rangle \) is a lower case Roman numeral, which is converted to a decimal value and the test is performed in the same way as the above decimal test.

6. The next pattern matches for \( \langle \text{prefix} \rangle \langle \text{sep} \rangle \langle n \rangle \) where \( \langle n \rangle \) is an upper case Roman numeral, which is converted to a decimal value and the test is performed in the same way as the above decimal test.

7. The next pattern matches for \( \langle \text{prefix} \rangle \langle \text{sep} \rangle \langle c \rangle \) where \( \langle c \rangle \) is either a lower case letter from a to z or an upper case letter from A to Z. The character is converted to its code point and the test is performed in the same way as the decimal pattern above.

8. If none of the above, the location aren’t considered consecutive.

Examples:

1. \texttt{\textbackslash glsextr\textbackslash record\{gls.sample\}\{\textbackslash page\}\{glsnumberformat\}\{1\}}
\texttt{\textbackslash glsextr\textbackslash record\{gls.sample\}\{\textbackslash page\}\{glsnumberformat\}\{2\}}

These records are consecutive. The prefix, counter and format are identical (so the test passes step 1), the locations match the decimal pattern and the test in step 4c passes.

2. \texttt{\textbackslash glsextr\textbackslash record\{gls.sample\}\{\textbackslash page\}\{glsnumberformat\}\{1\}}
\texttt{\textbackslash glsextr\textbackslash record\{gls.sample\}\{\textbackslash page\}\{textbf{2}\}}

These records aren’t consecutive since the formats are different.

3. \texttt{\textbackslash glsextr\textbackslash record\{gls.sample\}\{\textbackslash page\}\{glsnumberformat\}\{A.i\}}
\texttt{\textbackslash glsextr\textbackslash record\{gls.sample\}\{\textbackslash page\}\{glsnumberformat\}\{A.ii\}}

These records are consecutive. The prefix, counter and format are identical (so it passes step 1). The locations match the lower case Roman numeral pattern, where A is considered a prefix and the dot is consider a separator. The Roman numerals i and ii are
converted to decimal and the test is retried with the locations set to 1 and 2, respectively. This now passes the decimal pattern test (step 4c).

4. \texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{i.A}}
\texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{ii.A}}

These records aren’t consecutive. They match the alpha pattern. The first location is considered to consist of the prefix i, the separator \( . \) (dot) and the number given by the character code of A. The second location is considered to consist of the prefix ii, the separator \( . \) (dot) and the number given by the character code of A.

The test fails because the numbers are equal and the prefixes are different.

5. \texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{1.0}}
\texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{2.0}}

These records are consecutive. They match the decimal pattern, and then step 4a followed by step 4(a)iii. The \( .0 \) part is discarded and the test is retried with the first location set to 1 and the second location set to 2.

6. \texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{1.1}}
\texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{2.1}}

These records aren’t consecutive as the test branches off into step 4(a)i.

7. \texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{\@alph{1}}}
\texttt{\glslst{gls.sample}{}{page}{glsnumberformat}{\@alph{2}}}

These records are consecutive. The locations match the control sequence pattern. The control sequences are the same, so the test is retried with the first location set to 1 and the second location set to 2.

In this example, the location has been written to the file as \texttt{\@alph{⟨number⟩}} instead of fully expanding according to the normal behaviour of \texttt{\alph{⟨counter⟩}}. (Note that \texttt{\glslst{resourcefile} changes the category code of \@ to allow for internal commands in locations.) This unusual case is for illustrative purposes.

\texttt{\max-loc-diff=⟨value⟩}

This setting is used to determine whether two locations are considered consecutive. The value must be an integer greater than or equal to 1. (The default is 1.)

For two locations, \( ⟨\text{location-1}⟩ \) and \( ⟨\text{location-2}⟩ \), that have numeric values \( n_1 \) and \( n_2 \) (and identical prefix, counter and format), then the sequence \( ⟨\text{location-1}⟩, ⟨\text{location-2}⟩ \) is considered consecutive if

\[ 0 < n_2 - n_1 \leq \langle \text{max-loc-diff} \rangle \]

The default value of 1 means that \( ⟨\text{location-2}⟩ \) immediately follows \( ⟨\text{location-1}⟩ \) if \( n_2 = n_1 + 1 \).

For example, if \( ⟨\text{location-1}⟩ \) is “B” and \( ⟨\text{location-2}⟩ \) is “C”, then \( n_1 = 66 \) and \( n_2 = 67 \). Since \( n_2 = 67 = 66 + 1 = n_1 + 1 \) then \( ⟨\text{location-2}⟩ \) immediately follows \( ⟨\text{location-1}⟩ \).
This is used in the implicit range formations within the location lists (as described in the above section). So, for example, the list “1, 2, 3, 5, 7, 8, 10, 11, 12, 58, 59, 61” becomes “1–3, 5, 7, 8, 10–12, 58, 59, 61”.

The automatically indexing of commands like \gls means that the location lists can become long and ragged. You could deal with this by switching off the automatic indexing and only explicitly index pertinent use or you can adjust the value of max-loc-diff so that a range can be formed even if there are one or two gaps in it. By default, any ranges that have skipped gaps in this manner will be followed by \bibgls\passim. The default definition of this command is obtained from the resource file. For English, this is \passim (space followed by “passim”).

So with the above set of locations, if max-loc-diff={2} then the list becomes “1–12 passim, 58–61 passim” which now highlights that there are two blocks within the document related to that term.

\textbf{suffixF=⟨value⟩}

If set, an implicit range consisting of two consecutive locations ⟨loc-1⟩ and ⟨loc-2⟩ will be displayed in the location list as ⟨loc-1⟩⟨value⟩. This option doesn’t affect explicit ranges, even with merge-ranges={true}.

Note that suffixF={} sets the suffix to the empty string. To remove the suffix formation use suffixF={none}.

The default is suffixF={none}.

\textbf{suffixFF=⟨value⟩}

If set, an implicit range consisting of three or more consecutive locations ⟨loc-1⟩ and ⟨loc-2⟩ will be displayed in the location list as ⟨loc-1⟩⟨value⟩. This option doesn’t affect explicit ranges, even with merge-ranges={true}.

Note that suffixFF={} sets the suffix to the empty string. To remove the suffix formation use suffixFF={none}.

The default is suffixFF={none}.

\textbf{compact-ranges=⟨value⟩}

The ⟨value⟩ may be an integer ⟨n⟩ or false (equivalent to compact-ranges={0}) or true (equivalent to compact-ranges={3}). If no ⟨value⟩ is specified, true is assumed.

This setting allows location ranges such as 184–189 to appear more compactly as 184–9. The end location is encapsulated in the command \bibglscompact, so the range would actually become:

184\delimR\bibglscompact{digit}{18}{9}

If the location is in the form ⟨cs⟩{⟨loc⟩} (where ⟨cs⟩ is a command) then \bibglscompact will be inside the argument. For example, if the range would normally be:
5.10 Location List Options

\custom{184}\delimR\custom{189}
then it would become:
\custom{184}\delimR\custom{\bibglsc{\text{digit}}{18}{9}}

The numerical value given in compact-ranges=⟨⟨n⟩⟩ indicates that compaction should only occur if the actual location consists of at least ⟨n⟩ characters, for ⟨n⟩ ≥ 2. Any value of ⟨n⟩ less than 2 will switch off compaction.

For example, 189 consists of 3 characters, so it will be compacted with compact-ranges=\{3\} but not with compact-ranges=\{4\}. Whereas \custom{89} would only be compacted with compact-ranges=\{2\} because 89 only consists of 2 characters.

The compaction isn’t limited to decimal digits but it will only occur if both the start and end location have the same number of characters. For example, xvi–xviii can’t be compacted because the start consists of three characters and the end consists of five characters, whereas xxv–xxx can be compacted to xxv–x, which may look a little strange. In this case, you may want to consider changing the definition of \bibglsc{compact} so that it only performs the compaction for digits.

\textbf{see=⟨value⟩}

If an entry has a see field, this can be placed before or after the location list, or completely omitted (but the value will still be available in the see field for use with \glsxtrusesee). The required ⟨value⟩ must be one of:

- omit: omit the see reference from the location list.
- before: place the see reference before the location list.
- after: place the see reference after the location list (default).

The separator between the location list and the cross-reference is provided by \bibglssseesep. This separator is omitted if the location list is empty. The cross-reference is written to the location field using \bibglssusesee{⟨label⟩}.

\textbf{seealso=⟨value⟩}

This is like see but governs the location of the cross-references provided by the seealso field. You need at least v1.16 of glossaries-extra for this option. The values are the same as for see but the separator is given by \bibglssseelseasep. The cross-reference is written to the location field using \bibglssuseseealso{⟨label⟩}.

\textbf{alias=⟨value⟩}

This is like see but governs the location of the cross-references provided by the alias field. The separator is given by \bibglssaliassep. The cross-reference is written to the location field using \bibglssusealias{⟨label⟩}.
5.10 Location List Options

**alias-loc=⟨value⟩**

If an entry has an alias field, the location list may be retained or omitted or transferred to the target entry. The required ⟨value⟩ must be one of:

- keep: keep the location list;
- transfer: transfer the location list;
- omit: omit the location list.

The default setting is alias-loc={transfer}. In all cases, the target entry will be added to the see field of the entry with the alias field, unless it already has a see field (in which case the see value is left unchanged).

Note that with alias-loc={transfer}, both the aliased entry and the target entry must be in the same resource set. (That is, both entries have been selected by the same instance of \glsxtrresourcefile.) If you have glossaries-extra version 1.12, you may need to redefine \glsxtrsetaliasnoindex to do nothing if the location lists aren’t showing correctly with aliased entries. (This was corrected in version 1.13.)

**loc-prefix=⟨value⟩**

The loc-prefix setting indicates that the location lists should begin with \bibglslocprefix{⟨n⟩}. The ⟨value⟩ may be one of the following:

- false: don’t insert \bibglslocprefix{⟨n⟩} at the start of the location lists (default).
- {⟨prefix-1⟩},{⟨prefix-2⟩},…,{⟨prefix-n⟩}: insert \bibglslocprefix{⟨n⟩} (where ⟨n⟩ is the number of locations in the list) at the start of each location list and the definition of \bibglslocprefix will have an \ifcase condition:

\providecommand\bibglslocprefix[1]{{\% 
\ifcase#1
\or ⟨prefix-1⟩\bibglspostlocprefix 
\or ⟨prefix-2⟩\bibglspostlocprefix 
... 
\else ⟨prefix-n⟩\bibglspostlocprefix 
\fi }
}

- comma: equivalent to loc-prefix={, } but avoids confusion with the list syntax. That is, the prefix is a comma followed by a space for non-empty location lists.
- list: equivalent to loc-prefix={\pagelistname }.
• **true**: equivalent to \( \text{loc-prefix}={\bibglspagename,\bibglspagesname} \), where the definitions of \( \bibglspagename \) and \( \bibglspagesname \) are obtained from the \( \text{tag.page} \) and \( \text{tag.pages} \) entries in bib2gls's language resource file. This setting works best if the document's language matches the language file. However, you can redefine these commands within the document’s language hooks or in the glossary preamble.

If \( \langle \text{value} \rangle \) is omitted, **true** is assumed. The definition will be placed in the .glstex file according to \( \text{loc-prefix-def} \).

For example:

\[
\text{GlsXtrLoadResources}[\text{type}={\text{main}},\text{loc-prefix-def}={\text{individual}},\text{src}={\text{entries1}},\text{loc-prefix}={\text{false}}]
\]
\[
\text{GlsXtrLoadResources}[\text{type}={\text{main}},\text{loc-prefix-def}={\text{individual}},\text{src}={\text{entries2}},\text{loc-prefix}]
\]
\[
\text{GlsXtrLoadResources}[\text{type}={\text{symbols}},\text{src}={\text{entries3}},\text{loc-prefix}={p.,pp.}]
\]

This works since the conflicting \( \text{loc-prefix}={p.,pp.} \) and \( \text{loc-prefix}={\text{true}} \) are in different glossaries (assigned through the \( \text{type} \) key). The entries fetched from entries1.bib won't have a location prefix. The entries fetched from entries2.bib will have the location prefix obtained from the language resource file. The entries fetched from entries3.bib will have the location prefix "p." or "pp." (Note that using the \text{type} option isn't the same as setting the \text{type} field for each entry in the .bib file.)

If the \text{type} option isn’t used:

\[
\text{GlsXtrLoadResources}[\text{src}={\text{entries1}},\text{loc-prefix}={\text{false}}]
\]
\[
\text{GlsXtrLoadResources}[\text{src}={\text{entries2}},\text{loc-prefix}]
\]
\[
\text{GlsXtrLoadResources}[\text{src}={\text{entries3}},\text{loc-prefix}={p.,pp.}]
\]

then \( \text{loc-prefix}={\text{true}} \) takes precedence over \( \text{loc-prefix}={p.,pp.} \) (since it was used first). The entries fetched from entries1.bib still won’t have a location prefix, but the entries fetched from both entries2.bib and entries3.bib have the location prefixes obtained from the language resource file.

Note that if you identify some glossaries but not others (for example, you have dual entries in separate glossaries but only use \text{type} and not \text{dual-type}), then you will need to use \( \text{loc-prefix-def}={\text{global}} \) or \( \text{loc-prefix-def}={\text{local}} \).

\( \text{loc-prefix-def}=\langle \text{value} \rangle \)

This determines how the location prefix identified by \( \text{loc-prefix} \) is written to the .glstex file. The value may be one of:

• **global** the definition is globally defined using $\text{\providecommand}$;

• **local** the definition is locally defined using $\text{\providecommand}$ in the general glossary preamble ($\text{\glossarypreamble}$);
5.10 Location List Options

- **individual** the definition is locally defined using `\providecommand` in the glossary preamble of each type that has been identified in the current resource set, using options like `type` and `dual-type` (`\apptoglossarypreamble`).

The default is `loc-prefix-def={individual}`. Note that this can lead to an undefined control sequence error if locations appear in a glossary that hasn’t been detected by the resource set.

**loc-suffix=⟨value⟩**

This is similar to `loc-prefix` but there are some subtle differences. In this case `⟨value⟩` may either be the keyword `false` (in which case the location suffix is omitted) or a comma-separated list `⟨suffix-0⟩, ⟨suffix-1⟩, ..., ⟨suffix-n⟩` where `⟨suffix-0⟩` is the suffix to use when the location list only has a cross-reference with no locations, `⟨suffix-1⟩` is the suffix to use when the location list has one location (optionally with a cross-reference), and so on. The final `⟨suffix-n⟩` in the list is the suffix when the location list has `⟨n⟩` or more locations (optionally with a cross-reference).

This option will append `\bibglslocsuffix{⟨n⟩}` to location lists that either have a cross-reference or have at least one location. Unlike `\bibglslocprefix`, this command isn’t used when the location list is completely empty. Also, unlike `\bibglslocprefix`, this suffix command doesn’t have an equivalent to `\bibglspostlocprefix`.

If `⟨value⟩` omitted, `loc-suffix={\@.}` is assumed. The default is `loc-suffix={false}`.

The way the definition is written to the `.glstex` file is determined by `loc-suffix-def`. Note that if you identify some glossaries but not others (for example, you have dual entries in separate glossaries but only use `type` and not `dual-type`), then you will need to use `loc-suffix-def={global}` or `loc-suffix-def={local}`.

**loc-suffix-def=⟨value⟩**

This determines how the location suffix identified by `loc-suffix` is written to the `.glstex` file. The value may be one of:

- **global** the definition is globally defined using `\providecommand`;
- **local** the definition is locally defined using `\providecommand` in the general glossary preamble (`\glossarypreamble`);
- **individual** the definition is locally defined using `\providecommand` in the glossary preamble of each type that has been identified in the current resource set, using options like `type` and `dual-type` (`\apptoglossarypreamble`).

The default is `loc-suffix-def={individual}`. Note that this can lead to an undefined control sequence error if locations appear in a glossary that hasn’t been detected by the resource set.
loc-counters=⟨list⟩

Commands like \gls allow you to select a different counter to use for the location for that specific instance (overriding the default counter for the entry’s glossary type). This is done with the counter option. For example, consider the following document:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,style={tree}]{glossaries-extra}
\GlsXtrLoadResources[
  src={entries}% data in entries.bib
]
\begin{document}
\gls{pi}.
\begin{equation}
\gls[counter={equation}]{pi}
\end{equation}
\begin{equation}
\gls[counter={equation}]{pi}
\end{equation}
\newpage
\begin{equation}
\gls[counter={equation}]{pi}
\end{equation}
\newpage
\begin{equation}
\gls[counter={equation}]{pi}
\end{equation}
\newpage
\begin{equation}
\gls[counter={equation}]{pi}
\end{equation}
\newpage
\gls{pi}.
\newpage
\gls{pi}.
\newpage
\gls{pi}.
\newpage
\printunsrtglossaries
\end{document}

This results in the location list “1, 1–3, 3–5”. This looks a little odd and it may seem as though the implicit range formation hasn’t worked, but the locations are actually: page 1, equation 1,
5.10 Location List Options

equation 2, equation 3, page 3, page 4 and page 5. Ranges can’t be formed across different counters.

The `loc-counters={⟨list⟩}` option instructs bib2gls to group the locations according to the counters given in the comma-separated `⟨list⟩`. If a location has a counter that’s not listed in `⟨list⟩`, then the location is discarded.

For example:

```
\GlsXtrLoadResources[
  loc-counters={equation,page},% group locations by counter
  src={entries}% data in entries.bib
]
```

This will first list the locations for the equation counter and then the locations for the page counter. Each group of locations is encapsulated within the command `\bibglslocationgroup{(n)}{(counter)}{(locations)}`. The groups are separated by `\bibglslocationgroupsep`.

The `⟨list⟩` value must be non-empty. Use `loc-counters={as-use}` to restore the default behaviour, where the locations are listed in the document order of use, or `save-locations={false}` to omit the location lists. Note that you can’t form counter groups from supplemental location lists.

```
\save-index-counter=⟨value⟩
```

This option requires at least version 1.29 of glossaries-extra. The `⟨value⟩` may be one of:

- `false`: don’t create the `indexcounter` field (default);
- `true`: create the `indexcounter` field with the value set to the first `wrglossary` location;
- `⟨encap⟩`: create the `indexcounter` field with the value set to the first `wrglossary` location where the format is `⟨encap⟩`.

This setting will have no effect if the `indexcounter` package option hasn’t been used. In the case where the `⟨value⟩` is `⟨encap⟩`, make sure that this format takes priority in the location precedence rules (`--map-format`). If the location with that `⟨encap⟩` format value is discarded then it can’t be saved.

The `indexcounter` package option (glossaries-extra v1.29+) creates a new counter called `wrglossary` that’s incremented every time a term is indexed (recorded), except for cross-references such as `\glssee`. The increment is performed using `\refstepcounter` and is followed by `\label{wrglossary.(n)}` where `⟨n⟩` is the value of the `wrglossary` counter. This option is intended for use with the hyperref package to allow locations to link back to the particular part of the page where the term was referenced rather than to the top of the page.

Take care not to confuse this with the `indexed` special internal field introduced in glossaries-extra v1.49+. This is incremented on a per-entry basis and does not have an associated counter.
5.10 Location List Options

The **indexcounter** package option also automatically implements the option `counter={wrglossary}`, which means that each instance of `\gls{⟨id⟩}` writes the label information to the `.aux` file:

\newlabel{wrglossary.⟨n⟩}{⟨n⟩}{⟨page⟩}{wrglossary.⟨n⟩}{}

(where `⟨page⟩` is the page number) followed by the record:

\glsxtr@record{⟨id⟩}{wrglossary}{glsnumberformat}{⟨n⟩}

The location here is actually the value of the `wrglossary` counter not the page number, but `bib2gls` can pick up the corresponding `⟨page⟩` from the \newlabel command. It then replaces the record’s location `⟨n⟩` with:

\glsxtr@wrglossarylocation{⟨n⟩}{⟨page⟩}

(but it only does this for records that have the `wrglossary` counter).

The glossaries-extra package (v1.29+) adjusts the definition of `\glshypernumber` (which is internally used by `\glsnumberformat`, `\hyperbf` etc when `hyperref` has been loaded) so that if the counter is `wrglossary` then `\pageref` is used instead of `\hyperlink`. This means that the page number is displayed in the location list but it links back to the place where the corresponding `\label` occurred.

This method works partially with `makeindex` and `xindy` but from their point of view the location is the value of the `wrglossary` counter, which interferes with their ability to merge duplicate page numbers and form ranges. Since `bib2gls` is designed specifically to work with glossaries-extra, it’s aware of this special counter and will merge and collate the locations according to the corresponding `\label` occurred.

With the default `--merge-wrglossary-records` switch, if a term has multiple `wrglossary` records for a given page they will be merged. The reference link will be the dominant record for that page.

The **save-index-counter** option allows you to save the first of the `wrglossary` locations for a given entry or the first instance of a specific format of the `wrglossary` locations for a given entry. This location is stored in the `indexcounter` internal field using:

\GlsXtrSetField{⟨id⟩}{indexcounter}{\glsxtr@wrglossarylocation{⟨n⟩}{⟨page⟩}}

Since `\glsxtr@wrglossarylocation` simply expands to its first argument, the corresponding label can be obtained with:

\texttt{wrglossary.\glsxtr@wrglossarylocation{⟨n⟩}{⟨page⟩}}

For convenience, glossaries-extra-bib2gls provides:

\GlsXtrIndexCounterLink{⟨text⟩}{⟨label⟩}

which will do:
\hyperref[wrglossary.(value)]{(text)}

where \textit{(value)} is the value of the \texttt{indexcounter} field if it has been set. If the \texttt{indexcounter} field hasn’t been set (or if \texttt{hyperref} hasn’t been loaded) then just \textit{text} is done.

This provides a convenient way of encapsulating the \texttt{name} in the glossary so that it links back to the first \texttt{wrglossary} entry or the first \texttt{format=\{encap\} wrglossary} entry. This encapsulation can be done by providing a new glossary style or more simply by redefining \texttt{\glsnamefont}:

\begin{verbatim}
\renewcommand{\glsnamefont}[1]{%  
  \GlsXtrIndexCounterLink{#1}{\glscurrententrylabel}}
\end{verbatim}

Here’s a complete example:

\begin{verbatim}
\documentclass{article}
\usepackage{lipsum}% dummy filler text
\usepackage[colorlinks]{hyperref}
\usepackage[record,indexcounter]{glossaries-extra}
\newcommand{\primary}[1]{\hyperbf{#1}}
\GlsXtrLoadResources[
  src={entries},% terms defined in entries.bib
  save-index-counter={primary}
]
\renewcommand{\glsnamefont}[1]{%  
  \GlsXtrIndexCounterLink{#1}{\glscurrententrylabel}}
\begin{document}
A \gls{sample}. \lipsum*[1] A \gls{duck}.

An equation:
\begin{equation}
\gls[counter={equation}]{pi}
\end{equation}
\lipsum[2]

Another \gls[format={primary}]{sample}. \lipsum*[3] Another \gls{duck}.
\gls{pi}. \lipsum[4]
\end{document}
\end{verbatim}
5.11 Supplemental Locations

Note that the \texttt{counter={equation}} entry will have its own independent location. In this example, it’s difficult to tell the difference between 1 (the equation reference) and 1 (the page reference) in the location list for the \texttt{pi} entry.

The \texttt{format={primary}} instances indicate principal references. They’re displayed in bold (since \texttt{primary} is defined to use \texttt{\hyperbf}) and these are the locations saved in the \texttt{index-counter} field because that’s the \texttt{encap} identified by the \texttt{save-index-counter={primary}} setting.

5.11 Supplemental Locations

\textit{These options require at least version 1.14 of glossaries-extra.} If you require locations from multiple external sources, then you need at least version 1.36 of glossaries-extra (or, more specifically, glossaries-extra–bib2gls, which is automatically loaded by the \texttt{record={only}} package option).

The \texttt{glossaries-extra} package (from v1.14) provides a way of manually adding locations in supplemental documents through the use of the \texttt{thevalue} option in the optional argument of \texttt{\glsadd}. Setting values manually is inconvenient and can result in errors, so \texttt{bib2gls} provides a way of doing this automatically. Both the main document and the supplementary document need to use the \texttt{record} option. The entries provided in the \texttt{src} set must have the same labels as those used in the supplementary document. (The simplest way to achieve this is to ensure that both documents use the same \texttt{.bib} files and the same prefixes.)

For example, suppose the file \texttt{entries.bib} contains:

\begin{verbatim}
@entry{sample,
    name={sample},
    description="an example entry"
}

@abbreviation{html,
    short="html",
    long={hypertext markup language}
}

@abbreviation{ssi,
    short="ssi",
}
\end{verbatim}
Now suppose the supplementary document is contained in the file suppl.tex:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,counter={section}]{glossaries-extra}
\GlsXtrLoadResources[src={entries}]
\renewcommand{\thesection}{S\arabic{section}}
\renewcommand{\theHsection}{\thepart.\thesection}
\begin{document}
\part{Sample Part}
\section{Sample Section}
gls{goose}. \gls{sample}.
\part{Another Part}
\section{Another Section}
\gls{html}.
\gls{ssi}.
\printunsrtglossaries
\end{document}

This uses the section counter for the locations and has a prefix (\thepart.) for the section hyperlinks.

Now let's suppose I have another document called main.tex that uses the sample entry, but also needs to include the location (S1) from the supplementary document. The manual approached offered by glossaries-extra is quite cumbersome and requires setting the external location attribute and using \glsadd with the value={S1}, the H value={I.S1} and format =\{glsxtrsupphypernumber\}.

This can be simplified with bib2gls by using the supplemental-locations option, described below.

Version 1.36 of glossaries-extra-bib2gls introduces some special location formatting commands that don't use the externlocation attribute, but instead have an extra argument that indicates the external reference. The additional argument means that it can't be used by the format key, but with bib2gls you don't use \glsadd to record the external locations. Instead it obtains the records from the corresponding supplementary .aux file, and adjusts the location encapsulator as appropriate.
If bib2gls detects an older version of glossaries-extra, it will only allow one external supplemental source, and will set the externallocation attribute and use the glsxtrsupphyper-number format. Otherwise bib2gls will allow multiple sources and use the newer method.

\texttt{supplemental-locations=⟨basename⟩}

The value should be the base name (without the extension) of the supplementary document (\texttt{suppl} in the above example). If you have at least version 1.36 of glossaries-extra, the value may be a comma-separated list of base names (without the extensions) of the supplementary documents. If an older version is detected, bib2gls will issue a warning and only accept the first element of the list.

For example:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[
  supplemental-locations={suppl},% fetch records from suppl.aux
  src={entries}]
\begin{document}
\Gls{sample} document.
\printunsrtglossaries
\end{document}

The location list for \texttt{sample} will now be “1, S1” (page 1 from the main document and S1 from the supplementary document).

With glossaries-extra v1.36+, a regular location from the supplementary document will be encapsulated with:

\texttt{\glsxtrdisplaysupploc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨src⟩}{⟨location⟩}}

By default, this simply creates an external hyperlink to the supplementary document with the location as the hyperlink text. The hyperlink is created using \texttt{⟨src⟩} as the target path with the fragment part (anchor) formed from the prefix and location. The externallocation attribute is not set in this case. The actual formatting is done via:

\texttt{\glsxtrmultisupplocation{⟨location⟩}{⟨src⟩}{⟨format⟩}}

which ignores the \texttt{⟨format⟩} argument by default. Its definition is simply:

\texttt{\newcommand*{\glsxtrmultisupplocation}[3]{%}
  {% scope required to localise changes}
This locally sets the command \glsxtrsupplocationurl, which is checked by \glshypernumber to establish an external rather than internal link. You can redefine the supplemental location command to retain the original ENCAP used in the target document:

\renewcommand*{\glsxtrmultisupplocation}[3]{%  
% scope required to localise changes  
\def\glsxtrsupplocationurl{#2}%  
\csuse{#3}{#1}%
}

but remember that if a hyperlink is required, the identified control sequence name must correspond to a command that uses \glshyperlink (such as \hyperbf), otherwise you will lose the hyperlink.

With older versions of glossaries-extra, the original location format from the supplementary document will be replaced by \glxextrsupphypernumber, which again produces an external hyperlink. The externallocation attribute also needs to be set (this can be done automatically with supplemental-category) to identify the external document. The original format can't be accessed.

In both cases, if the document hasn’t loaded the hyperref package, the location will simply be displayed without a hyperlink. Even if both the main and the supplementary documents have loaded hyperref, note that not all PDF viewers can handle external hyperlinks, and some that can open the external PDF file may not recognise the destination within that file.

The special nameref locations (see --merge-nameref-on) are still identified with \glsxxtreldisplaylocnameref but the ⟨file⟩ argument will now be set.

As from bib2gls v1.7, any awkward characters in the file path are replaced with \bibglshrefchar or (for non-ASCII characters, when supported) \bibglshrefunicode. Both commands take two arguments: the hexadecimal character code and the actual character. In the case of \bibglshrefchar, the second argument is ignored, and the first is preceded by a literal percent character, so file name.pdf will be converted to:

file\bibglshrefchar{20}{ name.pdf
which will expand to file\%20name.pdf. In the case of \bibglshrefunicode, the first argument is ignored, so skráarnafn.pdf will be converted to:

skr\bibglshrefunicode{E1}{á}arnafn.pdf
which will expand to skráarnafn.pdf.

The supplementary location lists are encapsulated within \bibglssupplemental. With glossaries-extra v1.36+, this command will encapsulate the sub-lists with \bibglssupplementalsublist.
So the above example with an old version of glossaries-extra (pre 1.36) will set the supplemental location list (which only consists of one location) to:

\bibglissupplemental
\{1\}\{\setentrycounter[I]{section}\glsxtrsupphypernumber{S1}\

and the external target must be supplied through the externallocation attribute, which can be set with the supplemental-category option.

Whereas with at least version 1.36, the list will be:

\bibglissupplemental\{1\}\{\bibglissupplementalsublist\{1\}\{suppl.pdf\}\{\glsxtrdisplaysupploc[I]{section}\{glisnumberformat\{suppl.pdf\}\{S1\}\}}\

If an entry has both a main location list and a supplemental location list (such as the sample entry above), the lists will be separated by \bibglissupplementalsep. The sublists (when supported) are separated by \bibglissupplementalsubsep.

**supplemental-selection=(value)**

In the above example, only the sample entry is listed in the main document, even though the supplementary document also references the goose, html and ssi entries. By default, only those entries that are referenced in the main document will have supplementary locations added (if found in the supplementary document’s .aux file). You can additionally include other entries that are referenced in the supplementary document but not in the main document using supplemental-selection. The (value) may be one of the following:

- **all**: add all the entries in the supplementary document that have been defined in the .bib files listed in src for this resource set in the main document.
- **selected**: only add supplemental locations for entries that have already been selected by this resource set.
- **⟨label-1⟩,…,⟨label-2⟩**: in addition to all those entries that have already been selected by this resource set, also add the entries identified in the comma-separated list. If a label in this list doesn’t have a record in the supplementary document’s .aux file, it will be ignored.

Any records in the supplementary .aux file that aren’t defined by the current resource set (through the .bib files listed in src) will be ignored. Entry aliases aren’t taken into account when including supplementary locations.

For example:

```latex
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}
```
5.12 Sorting

Entries are typically displayed in an ordered list, but the glossaries-extra package is versatile enough to be used in wider contexts than simple terms, symbols or abbreviations. For example, entries could contain theorems or problems where the `name` supplies the title and the
description provides a description of the theorem or problem. Another field might then contain the proof or solution. Therefore, somewhat unusually for an indexing application, bib2gls also provides the option to shuffle the entries instead of sorting them.

This section covers the resource options for sorting primary entries. See section 5.14 for sorting dual entries and also sort-label-list for sorting field values that contain a comma-separated list of entry labels (such as the see or seealso fields).

The sort methods that use a comparison function (that is, all the sort methods except those listed in table 5.1) require a sort value for each entry. The function compares these values to determine the order. By default, this sort value is obtained from the sort field but for greater flexibility it’s best to not actually set this field. bib2gls has a set of fallbacks that it uses if a field it needs to access is missing. These fallbacks depend on the entry type and resource settings (see section 5.8).

For example, if a term defined with @index doesn’t have the sort field set then bib2gls will use the value given by the name field because name is the fallback field for sort for @index entries. If the name field isn’t set either then bib2gls will use the fallback for that field. In the case of @index that’s the entry’s label. If the sort field is explicitly set then there’s no need to use the fallback.

If, on the other hand, a term defined with @symbol doesn’t have the sort field set then bib2gls will use the value from the field identified by symbol-sort-fallback, which is the entry’s label by default (not the name field).

This means that if I don’t explicitly set the sort field for any entries then I can, for example, sort terms defined with @index by name and those defined with @symbol by description with the setting:

symbol-sort-fallback={description}

If the field used to obtain the sort value is changed (with sort-field) then the sort field won’t be queried. This reduces the flexibility of selecting the most appropriate field for given entry types. For example, sort-field={name} will force all entries to be sorted by the name field, which may not be appropriate for symbols.

If you choose a field whose value must be a label (such as parent or group) then the sort value will be that label.

You can have @preamble definitions that can be hidden from bib2gls’s interpreter. For example, no-interpret-preamble.bib might contain:

@preamble{"\providecommand{\sortop}[2]{#1 #2}"}

which is loaded using:

\GlsXtrLoadResources[src={no-interpret-preamble}, interpret-preamble={false}]

This provides a custom command:

\sortop{⟨text1⟩}{⟨text2⟩}
for internal use in the document. (Remember it won’t be defined on the first \LaTeX run before the .glistex file has been created and so is only used within entry fields.)

Another file, say, interpret-preamble.bib may provide a definition for \bib2gls:

\begin{verbatim}
@preamble{"\providecommand{\sortop}[2]{#2, #1}"
\end{verbatim}

which can be processed with:

\begin{verbatim}
\GlsXtrLoadResources[src={interpret-preamble}]
\end{verbatim}

to provide \bib2gls with this definition. The entries.bib file could contain:

\begin{verbatim}
@entry{caesar,
    name={\sortop{Gaius Julius}{Caesar}},
    first={Julius Caesar},
    text={Caesar},
    description={Roman politician and general}
}
\end{verbatim}

and then be processed with:

\begin{verbatim}
\GlsXtrLoadResources[src={entries}]
\end{verbatim}

The definition provided in interpret-preamble.bib, which swaps the two arguments around, is now picked up by \bib2gls, so the sort value becomes Caesar, Gaius Julius, but this new definition doesn’t affect the document since \LaTeX has already defined \sortop from the first resource set, so the name will appear as “Gaius Julius Caesar” in the glossary. (If you have \renewcommand rather than \providecommand, you can prevent the redefinition occurring in the document with write-preamble={false}.)

Alternatively both of these .bib files can be loaded in one resource set:

\begin{verbatim}
\GlsXtrLoadResources[src={interpret-preamble,entries}]
\end{verbatim}

Another possibility is to provide a custom package that contains the command definitions for the \bib2gls interpreter and load it with --custom-packages instead of having the interpret-preamble.bib file.

\texttt{sort=⟨value⟩}

The \texttt{sort} key indicates how primary entries should be sorted. If the \texttt{⟨value⟩} is omitted, \texttt{sort} =⟨resource⟩ is assumed. Note the differences between the keywords \texttt{resource}, \texttt{doc} and \texttt{locale}:

\textbf{resource} The default resource locale, which can be specified with the \texttt{locale} option. If that option hasn’t been set, then \texttt{resource} will be equivalent to \texttt{doc}. This option is new to \bib2gls v3.3. Previous versions had \texttt{sort=⟨doc⟩} as the default.

\textbf{doc} The document locale if it has been detected by tracklang. If no document language has been detected (or identified with --locale), then \texttt{doc} will be equivalent to \texttt{locale}.
locale The default Java locale.

The ⟨method⟩-reverse options reverse the result returned by the corresponding ⟨method⟩ comparator. However ⟨method⟩-reverse may not produce a list that’s the exact reverse of the underlying non-reversed ⟨method⟩ as the hierarchical structure or associated settings can affect the order.

No Sort Field

Most of the sort methods listed in table 5.1 don’t actually perform any sorting. This may cause a problem for hierarchical entries. In some cases this can lead to detached child entries or an attempt to define a child entry before its parent. The methods listed in this section all ignore the sort-field setting and all the various sort fallback settings, except where noted below.

- none (or unsrt): don’t sort the entries. (The entries will be in the order they were processed when parsing the data.)

If you need to order by definition but also maintain hierarchy then use:

```
save-definition-index,
sort-field={definitionindex},
sort={integer}
```

- random: shuffles rather than sorts the entries. This won’t work if there are hierarchical entries, so it’s best to use this option with flatten. The seed for the random generator can be set using shuffle (which also automatically sets sort={random} and flatten).

- use: order of use. This order is determined by the records written to the .aux file by the record package option. Dependencies and cross-references (including those identified with \glssee) come after entries with records.

Note that this is different from using the analogous option with makeindex or xindy, which does actually sort numerically, where each entry has an associated number set on the first use of that term that’s used as the sort value.

If you need to order by use but also maintain hierarchy then use:

```
save-use-index,
sort-field={useindex},
sort={integer}
```

- use-reverse: reverses the order that would be obtained with sort={use} without reference to hierarchy.
5.12 Sorting

Table 5.1: Summary of Available Sort Options: No Sort Field

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none or unsrt</td>
<td>don’t sort</td>
</tr>
<tr>
<td>random</td>
<td>shuffle entries</td>
</tr>
<tr>
<td>use</td>
<td>order of use</td>
</tr>
<tr>
<td>use-reverse</td>
<td>reverse order of use</td>
</tr>
<tr>
<td>recordcount†</td>
<td>order of record count</td>
</tr>
<tr>
<td>recordcount-reverse†</td>
<td>reverse order of record count</td>
</tr>
</tbody>
</table>

†Requires --record-count switch.

Table 5.2: Summary of Available Sort Options: Alphabet

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨lang tag⟩</td>
<td>sort according to this language tag</td>
</tr>
<tr>
<td>⟨lang tag⟩-reverse</td>
<td>reverse sort according to this language tag</td>
</tr>
<tr>
<td>resource</td>
<td>sort according to the default resource locale</td>
</tr>
<tr>
<td>resource-reverse</td>
<td>reverse sort according to the default resource locale</td>
</tr>
<tr>
<td>doc</td>
<td>sort according to the document locale</td>
</tr>
<tr>
<td>doc-reverse</td>
<td>reverse sort according to the document locale</td>
</tr>
<tr>
<td>locale</td>
<td>sort according to the default Java locale</td>
</tr>
<tr>
<td>locale-reverse</td>
<td>reverse sort according to the default Java locale</td>
</tr>
<tr>
<td>custom</td>
<td>sort according to sort-rule={⟨custom rule⟩}</td>
</tr>
<tr>
<td>custom-reverse</td>
<td>reverse sort according to sort-rule={⟨custom rule⟩}</td>
</tr>
</tbody>
</table>

Table 5.3: Summary of Available Sort Options: Letter (Non-Locale)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>letter-case</td>
<td>case-sensitive letter sort</td>
</tr>
<tr>
<td>letter-case-reverse</td>
<td>reverse case-sensitive letter sort</td>
</tr>
<tr>
<td>letter-nocase</td>
<td>case-insensitive letter sort</td>
</tr>
<tr>
<td>letter-nocase-reverse</td>
<td>reverse case-insensitive letter sort</td>
</tr>
<tr>
<td>letter-upperlower</td>
<td>upper-lower letter sort</td>
</tr>
<tr>
<td>letter-upperlower-reverse</td>
<td>reverse upper-lower letter sort</td>
</tr>
<tr>
<td>letter-lowerupper</td>
<td>lower-upper letter sort</td>
</tr>
<tr>
<td>letter-lowerupper-reverse</td>
<td>reverse lower-upper letter sort</td>
</tr>
</tbody>
</table>

Table 5.4: Summary of Available Sort Options: Letter-Number

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>letternumber-case</td>
<td>case-sensitive letter-number sort</td>
</tr>
<tr>
<td>letternumber-case-reverse</td>
<td>reverse case-sensitive letter-number sort</td>
</tr>
<tr>
<td>letternumber-nocase</td>
<td>case-insensitive letter-number sort</td>
</tr>
<tr>
<td>letternumber-nocase-reverse</td>
<td>reverse case-insensitive letter-number sort</td>
</tr>
<tr>
<td>letternumber-upperlower</td>
<td>upper-lower letter-number sort</td>
</tr>
<tr>
<td>letternumber-upperlower-reverse</td>
<td>reverse upper-lower letter-number sort</td>
</tr>
<tr>
<td>letternumber-lowerupper</td>
<td>lower-upper letter-number sort</td>
</tr>
<tr>
<td>letternumber-lowerupper-reverse</td>
<td>reverse lower-upper letter-number sort</td>
</tr>
</tbody>
</table>
### Table 5.5: Summary of Available Sort Options: Numerical

<table>
<thead>
<tr>
<th>Option</th>
<th>Sort Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>integer sort</td>
</tr>
<tr>
<td>integer-reverse</td>
<td>reverse integer sort</td>
</tr>
<tr>
<td>hex</td>
<td>hexadecimal sort</td>
</tr>
<tr>
<td>hex-reverse</td>
<td>reverse hexadecimal sort</td>
</tr>
<tr>
<td>octal</td>
<td>octal sort</td>
</tr>
<tr>
<td>octal-reverse</td>
<td>reverse octal sort</td>
</tr>
<tr>
<td>binary</td>
<td>binary sort</td>
</tr>
<tr>
<td>binary-reverse</td>
<td>reverse binary sort</td>
</tr>
<tr>
<td>float</td>
<td>float sort</td>
</tr>
<tr>
<td>float-reverse</td>
<td>reverse float sort</td>
</tr>
<tr>
<td>double</td>
<td>double sort</td>
</tr>
<tr>
<td>double-reverse</td>
<td>reverse double sort</td>
</tr>
<tr>
<td>numeric</td>
<td>locale-sensitive numeric sort</td>
</tr>
<tr>
<td>numeric-reverse</td>
<td>reverse locale-sensitive numeric sort</td>
</tr>
<tr>
<td>currency</td>
<td>locale-sensitive currency sort</td>
</tr>
<tr>
<td>currency-reverse</td>
<td>reverse locale-sensitive currency sort</td>
</tr>
<tr>
<td>percent</td>
<td>locale-sensitive percent sort</td>
</tr>
<tr>
<td>percent-reverse</td>
<td>reverse locale-sensitive percent sort</td>
</tr>
<tr>
<td>numberformat</td>
<td>locale-sensitive custom numeric sort</td>
</tr>
<tr>
<td>numberformat-reverse</td>
<td>reverse locale-sensitive custom numeric sort</td>
</tr>
</tbody>
</table>

### Table 5.6: Summary of Available Sort Options: Date-Time

<table>
<thead>
<tr>
<th>Option</th>
<th>Sort Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>locale-sensitive date sort</td>
</tr>
<tr>
<td>date-reverse</td>
<td>reverse locale-sensitive date sort</td>
</tr>
<tr>
<td>datetime</td>
<td>locale-sensitive date-time sort</td>
</tr>
<tr>
<td>datetime-reverse</td>
<td>reverse locale-sensitive date-time sort</td>
</tr>
<tr>
<td>time</td>
<td>locale-sensitive time sort</td>
</tr>
<tr>
<td>time-reverse</td>
<td>reverse locale-sensitive time sort</td>
</tr>
</tbody>
</table>
5.12 Sorting

- **recordcount**: order of record count (starting from 0). This order is determined by the total number of records written to the `.aux` file for each entry. Unlike the above methods, this performs a hierarchical sort. If letter groups are enabled with `--group`, this method will assign the entries to the number group.

  This option requires the `--record-count` switch. Although that switch makes `bib2gls` write the total record count to the `.glistex` file in the `recordcount` internal field (so that it can be accessed in the document), `bib2gls` doesn't actually have a field itself that contains the information. So although this option behaves much like `sort={integer}` it's not possible to select a field containing the required value. In the event of two or more entries having the same record count, the `identical-sort-action` option is used to determine the relative ordering between them.

- **recordcount-reverse**: reverse order of record count (ending with 0). All the above notes applying to `recordcount` also apply here.

Suppose the file `entries.bib` contains definitions of a set of symbols that don’t have any intuitive ordering (for example, they are all pictographs) then there may be no point in trying to order them, in which case you can do:

```latex
\GlsXtrLoadResources[src={entries},sort={none}]
```

Alternatively, you could list them in order of use:

```latex
\GlsXtrLoadResources[src={entries},sort={use}]
```

or by frequency of use. For example, starting with entries that don’t have any records followed by the least used entries (a rarely-used symbol may be harder to remember and most likely to be looked up in the glossary):

```latex
\GlsXtrLoadResources[src={entries},sort={recordcount}]
```

Or starting with the most used entries:

```latex
\GlsXtrLoadResources[src={entries},sort={recordcount-reverse}]
```

It all depends on what’s likely to be most useful to the reader.

Consider the following:

```latex
\newglossary*{frequent}{Most Frequently Used Terms}
\GlsXtrLoadResources[src={entries},sort={use},
  secondary={recordcount-reverse:frequent}]
\newcommand{\filterhook}[1]{%
  \GlsXtrIfFieldCmpNum*{recordcount}{#1}{>}{10}%
  {}%
  {\printunsrtglossaryskipentry}%
}
```

264
\begin{document}
\printunsrtglossary*[target={false},type={frequent}]{%
    \let\printunsrtglossaryentryprocesshook\filterhook
}
% Main body of the document ...
\printunsrtglossary
\end{document}

This has a summary at the start of the document that only contains entries that have at least
10 records and is ordered according to the total number of records (starting with the most
frequently used entry). The main glossary at the end of the document is ordered according
to use and contains all selected entries.

Compare this with the following:
\GlsXtrLoadResources[src={entries},sort={use}]
\newcommand{\filterhook}[1]{%  
    \GlsXtrIfFieldCmpNum*{recordcount}{#1}{>}{10}{}{\printunsrtglossaryskipentry}%
}{}
\begin{document}
\printunsrtglossary*[target={false},
title={Most Frequently Used Terms}]{%
    \let\printunsrtglossaryentryprocesshook\filterhook
}
% Main body of the document ...
\printunsrtglossary
\end{document}

This again has a summary at the start of the document that only contains entries that have
at least 10 records but is now ordered according to use.

Both examples assume there are no child entries as the filtering can cause parent entries
to be omitted. Both examples require \texttt{--record-count} but only the first example sorts ac-
cording to the record count.

\section*{Alphabet}

The sort methods listed in table 5.2 are for alphabets that are defined by a rule. These usually
ignore most punctuation and may ignore modifiers (such as accents). Use with \texttt{break-at}
to determine whether or not to split at word boundaries. The collation rules (except for the
custom options) are obtained from the locale provider (see page 28).

- \texttt{⟨lang tag⟩}: sort according to the rules of the locale given by the ietf language tag
  \texttt{⟨lang tag⟩}.
5.12 Sorting

- \langle lang \rangle-reverse: reverse sort according to the rules of the locale given by the IETF language tag \langle lang \rangle.

- resource: equivalent to sort=\{(lang tag)\} where \langle lang tag \rangle is obtained from the default resource locale.

- resource-reverse: equivalent to sort=\{(lang tag)-reverse\} where \langle lang tag \rangle is obtained from the default resource locale.

- locale: equivalent to sort=\{(lang tag)\} where \langle lang tag \rangle is obtained from the Java locale (which usually matches the operating system’s locale).

- locale-reverse: equivalent to sort=\{(lang tag)-reverse\} where \langle lang tag \rangle is obtained from the Java locale.

- doc: sort the entries according to the document locale. This is equivalent to sort=\{(lang tag)\} where \langle lang tag \rangle is the locale associated with the document language. In the case of a multi-lingual document, \langle lang tag \rangle is the locale of the last language resource file to be loaded through tracklang’s interface. It’s best to explicitly set the locale for multi-lingual documents to avoid confusion. If no document language has been set, this option is equivalent to sort=locale.

- doc-reverse: as doc but in reverse order.

- custom: sort the entries according to the rule provided by sort-rule.

- custom-reverse: reverse sort the entries according to the rule provided by sort-rule.

Note that sort=\{(lang tag)\} can provide more detail about the given locale than sort=\{doc\}, depending on how the document language has been specified. For example, with:

\documentclass{article}
\usepackage[ngerman]{babel}
\usepackage[record]{glossaries}
\GlsXtrLoadResources[src=german-terms]

the language tag will be de-1996, which doesn’t have an associated region, so this is equivalent to using sort=\{de-1996\}. Whereas with:

\documentclass[de-DE-1996]{article}
\usepackage[ngerman]{babel}
\usepackage[record]{glossaries}
\GlsXtrLoadResources[src=german-terms]

the language tag will be de-DE-1996 because tracklang has picked up the locale from the document class options, so this is equivalent to using sort=\{de-DE-1996\}. This is only
likely to cause a difference if a language has different sorting rules according to the region or if the language may be written in multiple scripts.

If no document locale has been set and the `locale` resource option hasn’t been used then the `sort={resource}` and `sort={doc}` will be equivalent to `sort={locale}`. For example, with:

\documentclass{article}
\usepackage[record]{glossaries}
\GlsXtrLoadResources[src={german-terms}]

the language tag will be whatever is the default locale for the JVM. For a user in Germany, this could be `de-DE-1996` and for a user in Austria this could be `de-AT-1996`.

A multilingual document will need to have the `sort` specified when loading the resource set to ensure the correct language is chosen. For example:

\GlsXtrLoadResources[src={english-terms},sort={en-GB}]
\GlsXtrLoadResources[src={german-terms},sort={de-DE-1996}]

Alternatively (as from bib2gls v3.3), use `locale`:

\GlsXtrLoadResources[locale={en-GB},src={english-terms}]
\GlsXtrLoadResources[locale={de-DE-1996},src={german-terms}]

**Letter (Non Locale)**

The sort methods listed in table 5.3 use letter comparators. These simply compare the character codes. The `nocase` options first convert the `sort` field to lower case before performing the sort to provide a case-insensitive comparison.

Punctuation isn’t ignored. Use `sort={⟨ lang tag ⟩}` with `break-at={none}` to emulate xindy’s locale letter ordering. The examples below show the ordering of the list antelope, bee, Africa, aardvark and Brazil.

- **letter-case**: case-sensitive letter sort. Upper case and lower case are in separate letter groups. Example:
  
  *Africa* (letter group upper case “A”), *Brazil* (letter group upper case “B”), *aardvark* (letter group lower case “a”), *antelope* (letter group lower case “a”), *bee* (letter group lower case “b”).

- **letter-case-reverse**: reverse case-sensitive letter sort. Example:
  
  *bee* (letter group lower case “b”), *antelope* (letter group lower case “a”), *aardvark* (letter group lower case “a”) *Brazil* (letter group upper case “B”), *Africa* (letter group upper case “A”).

- **letter-nocase**: case-insensitive letter sort. (All upper case characters will have first been converted to lower case in the sort value.) Example:
  
5.12 Sorting

- letter-nocase-reverse: reverse case-insensitive letter sort. Example:

- letter-upperlower: each character pair is first compared according to their lower case values. If these are equal, then they are compared according to case. This puts upper and lower case in the same letter group but the upper case comes first. Example:

- letter-upperlower-reverse: reverse upper-lower letter sort. This now puts the lower case letters first within the letter group. Example:

- letter-lowerupper: each character pair is first compared according to their lower case values. If these are equal, then they are compared according to case. This puts upper and lower case in the same letter group but the lower case comes first. Example:

- letter-lowerupper-reverse: reverse lower-upper letter sort. This now puts the upper case letters first within the letter group. Example:

**Letter-Number**

The sort methods listed in table 5.4 use a letter-integer hybrid. They behave in a similar way to the above letter sort methods, but if an integer number pattern is detected in the string then the sub-string containing the number will be compared. This only detects base 10 integers (unlike the numeric methods such as `sort={hexadecimal}` or `sort={float}`) but in addition to recognising all the digits in the Unicode “Number, Decimal Digit” category it also recognises the subscript and superscript digits, such as 1 (0x00B9) and 2 (0x00B2).

As with the letter sort methods, letters are compared using a character code comparison not by a locale alphabet. The closest locale-sensitive equivalent is to use `sort-number-pad` with a locale sort method.

For example, suppose the first string is `abc12foo` and the second string is `abc6bar`. Figure 5.1(a) shows the regular letter comparison using `sort={letter-case}`, where the subscript indicates the hexadecimal character code. The first three characters from each string are identical (abc). At this point there’s no difference detected, so the comparator moves on to the next character, 1₁₃₁ for the first string and 6₉₆ for the second string. Since 0x31 is less than 0x36, the first string (`abc12foo`) is considered less than the second (`abc6bar`).

268
5.12 Sorting

Figure 5.1: Regular letter comparison vs letter-number comparison. Comparing the strings `abc12foo` and `abc6bar`: (a) letter-case; (b) letter-number-case.

With the letter-number comparison using `sort={letternumber-case}`, the comparator starts in much the same way. The first three characters from each string are still identical, so the comparator moves on to the next character, 1 for the first string and 6 for the second. These are now both recognised as digits, so the comparator looks ahead and reads in any following digits (if present). For the first case, this is the sub-string 12 and, for the second case, 6 (figure 5.1(b)). These are both compared according to their integer representation $12 > 6$, so `abc12bar` is considered greater than `abc6foo` (that is, `abc12bar` comes after `abc6foo`).

The same result occurs for other numbering systems, for example if the Basic Latin digits 1, 2 and 6 are replaced with the corresponding Devanagari digits १, २ and ६. (But note that the letter comparisons will still be based on their Unicode values not according to a particular locale. This type of sort method is intended primarily for symbolic values, such as chemical formulae, rather than for words or phrases.)

Signed integers are also recognised, so `abc-12foo` is less than `abc+6bar`, which is again different from the result obtained with a straight letter comparator where the character `+` (0x2B) comes before the character `-` (0x2D). The sign must be followed by at least one digit for it to be recognised as a number otherwise it’s treated as a punctuation character.

If only one sub-string is numeric then the `letter-number-rule` is used to determine the result. Where both sub-strings are non-numeric, then the `letter-number-punc-rule` setting is used to determine the result according to the category of the characters, which may be one of the following:

- white space: belongs to the Unicode “Separator, Space” category. If both characters
are white space, then they are compared according to their Unicode values otherwise they are ordered according to the letter-number-punc-rule setting.

- letter: belongs to one of the Unicode categories “Letter, Uppercase”, “Letter, Lowercase”, “Letter, Titlecase”, “Letter, Modifier” or “Letter, Other”. If both characters are letters then, for sort method letternumber-⟨modifier⟩, the characters are compared in the same way as the corresponding letter-⟨modifier⟩ sort method otherwise they are ordered according to the letter-number-punc-rule setting.

- punctuation: everything else. If both characters are punctuation, then they are compared according to their Unicode value otherwise they are ordered according to the letter-number-punc-rule setting.

For simplicity, the actual sort value used during sorting isn’t a simple string but is converted into a list of objects that represent one of: letter, integer, space or other (punctuation). This reduces the amount of parsing of substrings that needs to be performed.

The examples below show the ordering of the list: \(\text{CH}_2\text{O}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{C}_5\text{H}_4\text{NCOOH}, \text{CO, Cl, Co, Co}_2\text{O}_3, \text{Co, CO}, \text{Co}\text{MoO}_4\) and \(\text{CoCl}_2\), for the setting letter-number-rule={between}, where the subscripts are the Unicode subscript characters.

- letternumber-case: case-sensitive letter-number sort. Example:
  \(\text{CH}_2\text{O}, \text{CO, CO}_2, \text{C}_5\text{H}_4\text{NCOOH, C}_{10}\text{H}_{10}\text{O}_4, \text{Cl, Co, CoCl}_2, \text{CoMoO}_4, \text{Co}_2, \text{Co}_2\text{O}_3\).
  (Order determined by: H < O < 5 < 10 < 1 < o.)

- letternumber-case-reverse: reverse case-sensitive letter-number sort. Example:
  \(\text{Co}_2\text{O}_3, \text{Co}_2, \text{CoMoO}_4, \text{CoCl}_2, \text{Co, Cl, C}_{10}\text{H}_{10}\text{O}_4, \text{C}_5\text{H}_4\text{NCOOH, CO, C}_2, \text{CH}_2\text{O}.\)

- letternumber-nocase: case-insensitive letter-number sort. The sort value is first converted to lower case. Note that letter-number-rule={between} doesn’t make sense in this context as there won’t be any upper case characters in the sort value, so numbers will always come before letters. Example:
  \(\text{C}_5\text{H}_4\text{NCOOH, C}_{10}\text{H}_{10}\text{O}_4, \text{CH}_2\text{O, CL, CO, CO}_2, \text{CO}_2, \text{CO}_2\text{O}_3, \text{COCl}_2, \text{C}\text{OMO}_4\).
  (Order determined by: 5 < 10 < h < 1 < o.)

- letternumber-nocase-reverse: reverse case-insensitive letter-number sort, so numbers will now always come after letters. Example:
  \(\text{C}\text{OMO}_4, \text{COCl}_2, \text{CO}_2\text{O}_3, \text{CO}_2, \text{CO}_2, \text{CO, CO, Cl, CH}_2\text{O, C}_{10}\text{H}_{10}\text{O}_4, \text{C}_5\text{H}_4\text{NCOOH}.\)

- letternumber-upperlower: upper-lower letter-number sort. This behaves slightly differently to letter-upperlower when used with letter-number-rule={between} and has a more complicated rule that’s determined by the character following the number and implied numbers inserted between letters. (There was a bug in earlier versions that has been corrected in v1.8 so you may find a slightly different ordering when upgrading.) Example:
CH₂O, C₂H₄NCOOH, C₁₀H₁₀O₄, Cl, CO, CO₂, Co, Co₂, CoCl₂, CoMoO₄, Co₂O₃.

(Order determined by: H < 5H < 10H < 1 < 0 < o, and for the terms starting with CO or Co: 2 comes after null and C < M < 20.)

Compare this with `letter-number-rule={before letter}` which results in the order:

C₅H₄NCOOH, C₁₀H₁₀O₄, CH₂O, Cl, CO, CO₂, Co, Co₂, Co₂O₃, CoCl₂, CoMoO₄.

• **letternumber-upperlower-reverse**: reverse upper-lower letter-number sort. Example (with `letter-number-rule={between}`):

Co₂O₃, CoMoO₄, CoCl₂, Co₂, Co, CO₂, CO, Cl, C₁₀H₁₀O₄, C₅H₄NCOOH, CH₂O.

Compare this with `letter-number-rule={before letter}` which results in the order:

CoMoO₄, CoCl₂, Co₂O₃, Co₂, Co, CO₂, CO, Cl, CH₂O, C₁₀H₁₀O₄, C₅H₄NCOOH.

Remember that the associated settings are reversed as well. So `letter-number-rule = {before letter}` results in numbers after letters.

• **letternumber-lowerupper**: lower-upper letter-number sort. As with the upper-lower option, this behaves slightly differently to `letternumber-lowerupper` when used with `letter-number-rule={between}` and has a more complicated rule. Example:

CH₂O, C₂H₄NCOOH, C₁₀H₁₀O₄, Cl, Co, Co₂, CoCl₂, CoMoO₄, Co₂O₃, CO, CO₂.

Compare this with `letter-number-rule={before letter}` which results in the order:

C₅H₄NCOOH, C₁₀H₁₀O₄, CH₂O, Cl, Co, Co₂, Co₂O₃, CoCl₂, CoMoO₄, CO, CO₂.

• **letternumber-lowerupper-reverse**: reverse lower-upper letter-number sort. Example (with `letter-number-rule={between}`):

CO₂, CO, Co₂O₃, CoMoO₄, CoCl₂, Co₂, Co, Cl, C₁₀H₁₀O₄, C₅H₄NCOOH, CH₂O.

**Numerical**

The sort methods listed in table 5.5 use numeric comparisons. The sort value is expected to be a numeric value. If it can’t be parsed then it’s treated as 0 (and a warning will be written to the transcript). These all recognise the digits in the Unicode “Number, Decimal Digit” category but, unlike the hybrid letter-number comparators above, they don’t recognise the superscript or subscript digits. The “non-locale” in some of the descriptions below indicates that the method doesn’t recognise locale-sensitive formatting, such as group separators.

• **integer**: integer sort. This is for non-locale integer sort values.

• **integer-reverse**: as above but reverses the order.

• **hex**: hexadecimal integer sort. This is for non-locale hexadecimal sort values.
5.12 Sorting

- **hex-reverse**: as above but reverses the order.
- **octal**: octal integer sort. This is for non-locale octal sort values.
- **octal-reverse**: as above but reverses the order.
- **binary**: binary integer sort. This is for non-locale binary sort values.
- **binary-reverse**: as above but reverses the order.
- **float**: single-precision sort. This is for non-locale decimal sort values.
- **float-reverse**: as above but reverses the order.
- **double**: double-precision sort. This is for non-locale decimal sort values.
- **double-reverse**: as above but reverses the order.
- **numeric**: locale-sensitive numeric sort. Use `numeric-locale` to set the locale.
- **numeric-reverse**: as above but reverses the order.
- **currency**: locale-sensitive currency sort. Use `numeric-locale` to set the locale.
- **currency-reverse**: as above but reverses the order.
- **percent**: locale-sensitive percent sort. Use `numeric-locale` to set the locale.
- **percent-reverse**: as above but reverses the order.
- **numberformat**: locale-sensitive custom numeric sort. Use `numeric-locale` to set the locale and `numeric-sort-pattern` to set the number pattern.
- **numberformat-reverse**: as above but reverses the order.

In general, it doesn't make much sense to have hierarchical entries that need to be sorted by a number, but it is possible as long as each level uses the same type of numbering.

**Date-Time**

The sort methods listed in table 5.6 are for dates and times. Use `date-sort-format` and `date-sort-locale` to specify the date format and locale.

- **date**: sort dates.
- **date-reverse**: as above but reverses the order.
- **datetime**: sort date and time information.
- **datetime-reverse**: as above but reverses the order.
5.12 Sorting

- **time**: sort times.
- **time-reverse**: as above but reverses the order.

If the field you want to sort by contains a date then the simplest way to sort is to ensure the date is in ISO format and then just use a letter sort. However it may be that the date is in the format particular to your locale or you have a mix of AD and BC. In which case you can use one of the date/time sort options (such as `sort={date}` or `sort={date-reverse}`). The locale is assumed to be your default locale (as given by the JVM) but if you are using a different locale this can be set with `date-sort-locale`. The pattern is assumed to be the default for that locale but you can change this with `date-sort-format`. If you provide your own custom pattern you must make sure that it matches the selected sort option.

Take care if you switch from using the JRE to the CLDR locale provider as you may find the default pattern changes.

The locale and pattern information is used by `bib2gls` to parse the field. If the field value can’t be parsed then `bib2gls` will issue a warning and assume the current date (or time).

The actual sort value that’s used by the comparator is numeric. In the case of the time-based `sort={datetime}` and `sort={time}` (or their `-reverse` versions), this value is the number of milliseconds since 1st January, 1970. In the case of `sort={date}` (or `sort={date-reverse}`), this value is obtained from \( a(y \times 10000 + m \times 100 + d) \) where \( y \) is the year, \( m \) is the month number, \( d \) is the day of month number, and \( a \) is an integer representation of the era (\(-1\) for BC and \(+1\) for AD).

Unlike the numeric sort methods (such as `sort={integer}`) the date-time sort methods set the sort field to a value that can be more easily parsed within the document and that should mostly achieve the same ordering if a letter comparator were to be used with it (except for BC dates, where the order needs to be reversed). This has the by-product of providing a field that you can access within the document that can be more easily parsed by BibTeX.

In general, it doesn’t make much sense to have hierarchical entries that need to be sorted by date, but it is possible as long as each level uses the same date format.

For example, suppose my `.bib` file contains:

```latex
@entry{journalentry,
   name={10 Jan 2017},
   description={an interesting journal entry}
}
```

The name field uses an abbreviated UK date format. If all my other entries also use this format in the name then I can sort them chronologically:

```latex
\GlsXtrLoadResources[
   src={entries},% data in entries.bib
   sort={date},
   date-sort-locale={en-GB},
   date-sort-format={medium}
]
```

273
5.12 Sorting

(The medium format is actually the default for this locale, and the locale matches my system locale, so I could omit both date-sort-locale and date-sort-format.)

If --verbose mode is on, the transcript will show the label, sort value and numeric value for each entry. In this case, the information is:

journalentry -> '+1 2017-01-10' [20170110]

The first value is the label (journalentry), the second value is assigned to the sort field (+1 2017-01-10) and the number in square brackets is the actual numeric value used by the comparator. The signed number at the start of the sort field +1 is the numeric representation of the era as used for the a variable in the computation of the numeric value (as described earlier).

If I change the format to date-sort-format={short}, then the date can’t be parsed correctly and bib2gls will issue the following warning:

Warning: Can’t parse sort value '10 Jan 2017' for 'journalentry'
(pattern: 'dd/MM/yyyy')

This shows the value that bib2gls is trying to parse (10 Jan 2017) for the entry identified by the given label (journalentry). The pattern bib2gls expects is also given (dd/MM/yyyy).

shuffle={seed}

Automatically sets sort={random} and flatten. The value ⟨seed⟩ may be omitted. If present, it should be an integer used as a seed for the random number generator.

sort-field={field}

The sort-field key indicates which field provides the sort value. The ⟨field⟩ must be a recognised field name or you may use sort-field={id} to sort according to the label. The default value is the sort field (which is typically inferred rather than explicitly set).

Example:

\GlsXtrLoadResources[
  src={entries-terms},% data in entries-terms.bib
  sort-field={symbol},% sort by 'symbol' field
  sort={letter-case}% case-sensitive letter sort
]

This sorts the entries according to the symbol field using a case-sensitive letter comparison.

In general it’s better to use the default sort-field={sort} and adjust the fallbacks instead (see section 5.8). The sort-field option is provided if you want to use a specific field regardless of the entry type.
If an entry is missing a value for \textit{⟨field⟩}, then the value of the fallback field will be used instead. If \texttt{missing-sort-fallback} is set, then that’s used as the fallback, otherwise it depends on the entry type. If no fallback field can be found, the entry’s label will be used.

For the specific case with the default \texttt{sort-field=⟨sort⟩} setting, the fallback for the \texttt{sort} field is governed not only by the entry type but also by some associated settings:

- If the entry’s original type (before being aliased with \texttt{entry-type-aliases}) is identified in \texttt{custom-sort-fallbacks}, then if the \texttt{sort} field is missing the value is obtained from the supplied custom mapping.

- If the entry is defined using \texttt{@entry} (or a dual form that acts like \texttt{@entry}), then if the \texttt{sort} field is missing the value is obtained from the field identified by \texttt{entry-sort-fallback}. If that field is also missing then that field’s fallback is used.

- For the index entry types like \texttt{@index} or \texttt{@indexplural}, then if the \texttt{sort} field is missing the value is obtained from the \textit{name} field. If that field is also missing, then the value is obtained from the particular entry type’s fallback for the \textit{name} field. (For example, the entry’s label for \texttt{@index} or the \textit{plural} field for \texttt{@indexplural}.)

- If the entry is defined with an abbreviation type (for example, \texttt{@abbreviation} or \texttt{@acronym}) then if the \texttt{sort} field is missing, \texttt{bib2gls} will fallback on the field given by \texttt{abbreviation-sort-fallback}.

- The symbol-like entry types fallback on the field given by \texttt{symbol-sort-fallback} if the \texttt{sort} field is missing.

- Entries defined using \texttt{@bibtexentry} fallback on the field given by \texttt{bibtexentry-sort-fallback}, which defaults to the \textit{name} field. Note that this only applies to the main entry. The spawned \texttt{@contributor} entries behave like \texttt{@index}.

Use \texttt{dual-sort-field} when sorting dual entries.

\begin{verbatim}
missing-sort-fallback=⟨field⟩
\end{verbatim}

With \texttt{sort-field=⟨⟨sort-field⟩⟩}, if the value of the field identified by \texttt{⟨sort-field⟩} is missing, then \texttt{bib2gls} behaves as follows:

1. If \texttt{missing-sort-fallback=⟨⟨fallback-field⟩⟩} is set, then \texttt{bib2gls} will fallback on the value provided by the field \texttt{⟨fallback-field⟩}. If \texttt{⟨fallback-field⟩} is missing, then \texttt{bib2gls} will query the entry type’s fallback for \texttt{⟨fallback-field⟩} (not for \texttt{⟨sort-field⟩}). The \texttt{⟨fallback-field⟩} must be a known field but not an internal field. It can’t be the \texttt{sort} field. (Take care not to cause an infinite loop if \texttt{sort-field} has been changed.) Unlike the other sort fallback options, such as \texttt{entry-sort-fallback}, the \texttt{⟨fallback-field⟩} can’t be a keyword (to identify the label) and can’t be a composite.

2. If the entry type has a fallback rule for \texttt{⟨sort-field⟩}, then that rule is used (see section 5.8). When \texttt{sort-field=⟨sort⟩} this means:
5.12 Sorting

- If the entry’s original entry type has been identified in `custom-sort-fallbacks`, then `bib2gls` will fallback on the designated custom setting.

- If the entry was defined using one of the index types (such as `@index`), then `bib2gls` will fallback on the `name` field.

- If the entry was defined using the `@entry` type (or a dual form that acts like `@entry`), then `bib2gls` will fallback on the field given by `entry-sort-fallback`.

- If the entry was defined using one of the symbol types (such as `@symbol`), then `bib2gls` will fallback on the field given by `symbol-sort-fallback`.

- If the entry was defined using one of the abbreviation types (such as `@abbreviation`), then `bib2gls` will fallback on the field given by `abbreviation-sort-fallback`.

- If the entry was defined using `@bibtexentry` (but not the spawned `@contributor` entries), then `bib2gls` will fallback on the field given by `bibtexentry-sort-fallback`.

If ⟨sort-field⟩ is not `sort`, then there may not be a fallback, in which case the next condition applies:

3. Otherwise the sort value will be set to the entry label and `bib2gls` will issue a warning.

The default setting is `missing-sort-fallback={}`, which means that step 1 above is omitted.

Use `dual-missing-sort-fallback` when sorting dual entries separately from primaries, and use `secondary-missing-sort-fallback` for secondary sorting.

`trim-sort=(boolean)`

If the interpreter is used to determine the sort value, this setting governs whether or not the interpreter should trim leading and trailing spaces. The default setting is `trim-sort={true}`.

This option automatically sets `dual-trim-sort={⟨boolean⟩}` and `secondary-trim-sort={⟨boolean⟩}`.

`sort-replace=⟨list⟩`

This option may be used to perform regular expression substitutions on the sort value and has the same syntax as `labelify-replace`. The value is required for this key but may be empty, which indicates that the setting is switched off.

This action is done after the interpreter parses the sort value (if applicable) and before `sort-number-pad` (if applicable). For example, suppose the sort value is:

\[\approx 3.14\]

then the interpreter will convert this to \(\approx 3.14\) but:

`sort-replace={⟨\text{\glshex2248}⟩}`
can be used to strip the \( \approx \) symbol (0x2248) so that the value can now be parsed as a number if \texttt{sort=\{double\}} has been used.

Use \texttt{dual-sort-replace} for dual and \texttt{secondary-sort-replace} for secondary sort methods.

**sort-rule=\{value\}**

If the \texttt{sort=\{custom\}} option is used, the sort rule must be provided with \texttt{sort-rule}. If \texttt{sort} is not set to custom, the \texttt{sort-rule} setting will be ignored. This setting uses Java’s \texttt{RuleBasedCollator} class [6], and the rule syntax needs to conform to that format.

Remember that the options will be expanded as they are written to the .\texttt{aux} file, so be careful of any special characters that occur in the rule. For the special characters # % _ & { and } you can use \#, \%, _, & and { and }. These will be written to the .\texttt{aux} file with the leading backslash, but \texttt{bib2gls} will remove it for this resource option. Remember that the \texttt{glossaries} package provides \texttt{\glsbackslash} and \texttt{\glstildechar} which can be used to produce a literal backslash (\) and tilde (\-).

You can also use \texttt{\string\u\langle hex\rangle} (where \langle hex\rangle is a hexadecimal code) to represent a Unicode character. For example:

\begin{verbatim}
\GlsXtrLoadResources[
  sort=\{custom\},
  sort-rule=\{< a,A < b,B < c,C < ch,CH < d,D
     < dd,Dd,DD < e,E < f,F < ff,Ff,FF
     < g,G < ng,Ng,NG < h,H < ij,Ij,IJ
     < i,I < j,J < k,K < l,L < ll,Ll,LL < m,M
     < n,N < o,O < p,P < ph,Ph,PH < q,Q < r,R < rh,Rh,RH
     < s,S < t,T < th,TH,TH < u,u,U < v,V < w,W < x,X < y,Y < z,Z
     \langle string\u\langle hex\rangle\}\}
]
\end{verbatim}

It’s best to use \texttt{\string} rather than \texttt{\protect} to avoid unwanted spaces interfering with \langle hex\rangle. Note that \texttt{glossaries-extra v1.21+} provides \texttt{\glshex} which just does \texttt{\string\u} so you can do \texttt{\glshex 00E6} instead of \texttt{\string\u00E6}. This is only one character different, but you can redefine \texttt{\glsxtrresourceinit} to locally set \texttt{\u} to \texttt{\glshex} while the protected write is performed. For example:

\begin{verbatim}
\renewcommand*{\{\glsxtrresourceinit\}\{\let\u\glshex\}
\end{verbatim}

Then you can just do \texttt{\u00E6} instead of \texttt{\string\u00E6}. Note that \texttt{\GlsXtrResourceInitEscSequences} performs a similar assignment, so you can instead do:

\begin{verbatim}
\renewcommand*{\{\glsxtrresourceinit\}\{%
  \GlsXtrResourceInitEscSequences
}\}
\end{verbatim}

\footnote{The command definition was moved to glossaries-extra-bib2gls from version 1.27 since it’s only needed with \texttt{bib2gls}.}
5.12 Sorting

The glossaries-extra-bib2gls package (which is automatically loaded by the record option) provides some commands for common rule blocks that may be used in the construction of custom rules. For example:

\begin{verbatim}
sort-rule={\glsxtrcontrolrules
 ;\glsxtrsparserules
 ;\glsxtrnonprintablerules
 ;\glsxtrcombiningdiacriticrules
 ,\glsxtrhyphenrules
 <\glsxtrgeneralpuncrules
 <\glsxtrdigitrules
 <\glsxtrfractionrules
 <\glsxtrMathItalicGreekIrules
 <\glsxtrGeneralLatinIVrules
 <\glsxtrLatinAA
 <\glsxtrLatinOslash
 }
\end{verbatim}

This places the Greek maths symbols (such as $\alpha$) before the Latin block. See the glossaries-extra documentation for further details of these commands.

You might find it convenient to provide similar commands in a package for rules you may often need. For example, suppose I have a package called, say, mapsymbols for providing map symbols:

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}
\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
 < \glshex 2693 % anchor 0x2693
 < \glshex 26BD % football 0x26BD
}
\end{verbatim}

In addition to mapsymbols.sty, I also need to create mapsymbols.bib to provide the appropriate definitions for bib2gls:

\begin{verbatim}
@preamble{"\glsxtrprovidecommand{\Harbour}{\char"2693}
 \glsxtrprovidecommand{\Battlefield}{\char"2694}
 \glsxtrprovidecommand{\Stadium}{\char"26BD}"
\end{verbatim}
5.12 Sorting

The use of \glsxtrprovidecommand will override any previous definitions of these commands in bib2gls’s interpreter but will act like \providecommand within the document, and so won’t interfere with the commands defined in mapsymbols.sty. Now I can just do:

\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  
  ;\glsxtrspacerules  
  ;\glsxtrnonprintablerules  
  ;\glsxtrcombiningdiacriticrules  
  ;\glsxtrhyphenrules  
  \MapSymbolOrder % <--- custom map symbols  
  ;\glsxtrgeneralpuncrules  
  ;\glsxtrdigitrules  
  ;\glsxtrfractionrules  
  <\glsxtrMathItalicGreekIrules  
  <\glsxtrGeneralLatinIrules  
}

An alternative to providing mapsymbols.bib is to provide a custom package just for bib2gls’ use. For example, mapsymbols-bib2gls.sty:

% Provided for bib2gls only.  
% Use \usepackage{mapsymbols} in the document.  
\NeedsTeXFormat{LaTeX2e}  
\ProvidesPackage{mapsymbols-bib2gls}  
\glsxtrprovidecommand\Harbour{\char"2693}  
\glsxtrprovidecommand\Battlefield{\char"2694}  
\glsxtrprovidecommand\Stadium{\char"26BD}  
\endinput

and instruct bib2gls to parse it with --custom-packages mapsymbols-bib2gls (and use mapsymbols.sty in the document). Remember that bib2gls isn’t a \TeX engine so make sure to only use simple commands in this file.

\texttt{break-at=(option)}

This option automatically implements \texttt{dual-break-at=(option)} and \texttt{secondary-break-at=(option)}.
5.12 Sorting

The alphabet sort options (table 5.2) typically list non-letter characters before alphabetical characters and spaces are quite often in the ignored set. This means that the alphabet sort options are naturally in a letter order, similar to xindy’s ord/letorder module. (This isn’t the same as \texttt{sort=\{letter-nocase\}}, which just sorts according to the Unicode value not according to a particular alphabet.)

In order to replicate \texttt{makeindex} and xindy’s default word order, \texttt{bib2gls} splits up the sort value at word boundaries and inserts a marker (identified by \texttt{break-marker}). For example, if the sort value is “sea lion” then it’s actually converted to \texttt{sea|lion|} whereas “sea” becomes \texttt{sea|} and “seal” becomes \texttt{seal|}. The default marker is \texttt{|} which is commonly placed in collation rules before digits but after the ignored characters, such as spaces and hyphens.

Note that this action removes non-letters, so for example, if the sort value is \# (parameter) then it will be converted to \texttt{parameter|} (hash, space and parentheses removed). If you only want to break at spaces (optionally following a comma) use the following instead:

\begin{verbatim}
break-at={none},
sort-replace={{{,? +}{|}}}
\end{verbatim}

You can change the construction of the break points with \texttt{break-at=}\texttt{\{\langle option\rangle\}} where \texttt{\langle option\rangle} may be one of:

- word: break at word boundaries (default). Note that what constitutes a word varies according to the locale but usually anything that’s not alphanumeric will designate a word-boundary. The characters between words are discarded. For example, the sort value “Tom, Dick, and Harry” becomes \texttt{Tom|Dick|and|Harry}, which has discarded the comma and space characters.

- character: break after each character.

- sentence: break after each sentence.

- upper-notlower: break after any upper case character that’s not followed by a lower case character. For example, “MathML” becomes \texttt{MathM|L|} and “W3C” becomes \texttt{W|3C|}.

- upper-upper: break after any upper case character that’s followed by an upper case character.

- upper-notlower-word: first applies break-points according to upper-notlower and then according to word.

- upper-upper-word: first applies break-points according to upper-upper and then according to word.

- none: don’t create break points. Use this option to emulate \texttt{makeindex} or xindy’s letter ordering, or combine with \texttt{sort-replace} to insert custom break points.

This option is ignored when used with the non-alphabetic \texttt{sort} options. You can find the break points in the \texttt{sort} field for the entry’s definition in the \texttt{.glistex} file (which is provided
for information rather than for use in the document). Alternatively, use the --debug switch to show the break points in the transcript. (This will also show the collation rule.)

If you want to selectively apply break points only to certain entries, use break-at-match or break-at-not-match.

**break-marker={marker}**

This option automatically implements the dual and secondary settings dual-break-marker ={{marker}} and secondary-break-marker={{marker}}.

The break marker can be changed using break-marker={{marker}}, where ⟨marker⟩ is the character to use. For example, break-marker={-} will use a hyphen. The marker may be empty, which effectively strips the inter-word punctuation. For example, with break-marker={}, “Tom, Dick, and Harry” becomes TomDickandHarry and “sea lion” simply becomes sealion. If ⟨marker⟩ is omitted, break-marker={} is assumed.

**break-at-match={key=value list}**

This option automatically implements dual-break-at-match={⟨option⟩} and secondary-break-at-match={⟨option⟩}.

If you have break-at set to create break points (for example, with break-at={word}) then you can specify which entries should have break points with this option. The value has the same syntax as match. If an entry matches the criteria, then break points are added, otherwise no break points are added. For example, to only have break points for entries defined with @index or @indexplural:

break-at-match={entrytype=index(plural)?}

This option has no effect with break-at={none}.

**break-at-match-op={value}**

This option automatically implements dual-break-at-match-op={⟨option⟩} and secondary-break-at-match-op={⟨option⟩}.

If the value of break-at-match contains more than one ⟨key⟩=⟨pattern⟩ element, the break-at-match-op determines whether to apply a logical AND or a logical OR. The ⟨value⟩ may be either and or or. The default is break-at-match-op={and}.

**break-at-not-match={key=value list}**

This option automatically implements dual-break-at-not-match={⟨option⟩} and secondary-break-at-not-match={⟨option⟩}.

For example, to prevent entries defined with @symbol from having break points:

break-at-not-match={entrytype=symbol}

Like break-at-match but negates the match. This option has no effect with break-at={none}. 

281
sort-number-pad=⟨number⟩

This option automatically implements the dual and secondary settings dual-sort-number-pad=⟨number⟩, secondary-sort-number-pad=⟨number⟩.

If ⟨number⟩ is greater than 1, any integer sub-strings found in the sort value will be zero-padded up to this value. Since the - character is often ignored by rule-based sort methods, any signs found will be replaced with the markers given by sort-pad-plus and sort-pad-minus, which should be chosen to ensure that negative numbers are ordered before positive numbers (if this is desired). An unsigned number will have the sort-pad-plus marker inserted before it. The default value is sort-number-pad={0}, which doesn’t implement any padding.

If you use this with a locale sort method, it’s best to also set break-at={none}, as the default word boundary break points will likely be confused by a mix of alphanumerics.

sort-pad-plus=⟨marker⟩

This option automatically implements the dual and secondary settings dual-sort-pad-plus=⟨marker⟩, secondary-sort-pad-plus=⟨marker⟩.

This option only has an effect when used with sort-number-pad=⟨number⟩ where ⟨number⟩ is greater than 1. Positive numbers will have their sign replaced with ⟨marker⟩. The default setting is sort-pad-plus={>}.

sort-pad-minus=⟨marker⟩

This option automatically implements the dual and secondary settings dual-sort-pad-minus=⟨marker⟩, secondary-sort-pad-minus=⟨marker⟩.

This option only has an effect when used with sort-number-pad=⟨number⟩ where ⟨number⟩ is greater than 1. Negative numbers will have their sign replaced with ⟨marker⟩. The default setting is sort-pad-plus={<}.

identical-sort-action=⟨value⟩

This option automatically implements the dual and secondary settings dual-identical-sort-action=⟨value⟩ and secondary-identical-sort-action=⟨value⟩.

This option determines what the comparator should do if two entries at the same hierarchical level are considered equal. The ⟨value⟩ may be one of:

- none: don’t take any further action if sort values are identical;
- def if sort values are identical, order them according to definition;
- use: if sort values are identical, order them according to use in the document (order determine by a normal record);
- id: if sort values are identical, compare the entry labels;
5.12 Sorting

- original id: if sort values are identical, compare the original unprefixed entry labels (as given in the .bib file);
- \(\langle field\rangle\): if sort values are identical, compare the values from the given \(\langle field\rangle\).

For the last three cases, a simple case-sensitive string comparison is used. If \(\langle value\rangle\) isn’t a recognised keyword or valid field an error will occur. The default setting is \texttt{identical-sort-action=\{id\}}. If you’re using one of the sort rules listed in table 5.2 and you also want a locale-sensitive sort used on the fallback, then you need to use \texttt{sort-suffix} instead.

\texttt{bib2gls} allows duplicate sort values, but this can cause a problem for hierarchical entries where parent entries with duplicate sort fields are clumped together and their children follow. To prevent this from happening, the \texttt{identical-sort-action=\{id\}} setting will fallback on comparing the labels. Since all labels must be unique, this means comparisons between two different entries are all either strictly higher or strictly lower.

This action occurs after any suffixes have been appended through \texttt{sort-suffix}.

\texttt{sort-suffix=\{value\}}

This option automatically implements the dual and secondary settings \texttt{dual-sort-suffix=\{\langle value\rangle\}} and \texttt{secondary-sort-suffix=\{\langle value\rangle\}}. The value may be one of:

- none: don’t append a suffix to any \texttt{sort} value;
- non-unique: append a numeric suffix to non-unique \texttt{sort} values;
- \(\langle field\rangle\): append the value of the given field (if set) to the \texttt{sort} field. The given field must be defined (has an associated key for use in \texttt{\newglossaryentry}) but may be unset. If the interpreter is on, the field contents will be interpreted. If the field is just a label (such as the \texttt{category} field) you may find it simpler to use \texttt{identical-sort-action=\{\langle field\rangle\}} instead.

The default setting is \texttt{sort-suffix=\{none\}}.

This option only affects the alphabetic (table 5.2), letter (table 5.3) and letter-number (table 5.4) sort rules. For the other types of sort methods (not including the no-sort options listed in table 5.1) you’ll need to use \texttt{identical-sort-action} to prevent problems occurring with duplicate sort values.

In the case of \texttt{sort-suffix=\{non-unique\}}, this will only append a suffix to the duplicate sort values (within the same hierarchical level). The first sort value to be encountered isn’t given a suffix.

The \texttt{sort-suffix=\{\langle field\rangle\}} setting will only append a suffix if that field is set, but (if set) it will apply the suffix to all \texttt{sort} values, even those that are unique.

If you use \texttt{--verbose}, then \texttt{bib2gls} will write information in the transcript when it appends a suffix to the sort value. The message:

Sort value \'\langle sort\rangle\' (entry \'\langle id\rangle\') not unique for the entry's hierarchical level.

283
indicates that an entry with the given \textlangle sort\rangle value has already been found within the same hierarchical level as the currently processed entry (whose label is given by \textlangle id\rangle). The same hierarchical level in this context means that either both entries don't have a parent or both entries have the same parent. (That is, the entries are considered siblings.)

This message will then be followed by:

Appending suffix \textlangle suffix\rangle to the sort value \textlangle sort\rangle for entry \textlangle id\rangle.

which indicates that the entry (identified by the label \textlangle id\rangle) has been assigned the sort value given by \textlangle sort\rangle\langle suffix\rangle. If any break markers are applied, this is done after the suffix has been appended.

For example, suppose in my document I want to write about \texttt{makeglossaries} (the application) and \texttt{makeglossaries} (the command). I might decide to define semantic commands:

\begin{verbatim}
\newcommand*{\application}[1]{\texttt{#1}}
\newcommand*{\command}[1]{\texttt{\glsbackslash #1}}
\end{verbatim}

In my .bib file I might have:

\begin{verbatim}
@entry{cs.makeglossaries,
  name={\command{makeglossaries}},
  category={command},
  description={opens glossary files}
}

@entry{ap.makeglossaries,
  name={\application{makeglossaries}},
  category={application},
  description={Perl script}
}
\end{verbatim}

If \texttt{bib2gls} is provided with the definitions of \texttt{application} and \texttt{command} (by interpreting the \texttt{@preamble} or a package provided with --custom-packages) then it will determine that the sort value for \texttt{cs.makeglossaries} is \texttt{makeglossaries} and the sort value for \texttt{ap.makeglossaries} is just \texttt{makeglossaries}. These are two distinct sort values from \texttt{bib2gls}'s point of view although the sort rule may consider them identical if the rule ignores the \texttt{"} character (such as the locale sort methods), in which case, \texttt{bib2gls} will then act according to identical-sort-action.

If \texttt{bib2gls} isn't provided with these custom definitions, then it will ignore those semantic commands and both entries will end up with the sort value \texttt{makeglossaries}. The second instance will be recognised as a duplicate and the sort value will be converted to \texttt{makeglossaries1} (where the automated suffix is 1 and the suffix marker, see below, is the empty string). Whereas with, say, \texttt{sort-suffix-marker=\{.\}} then the sort value would become \texttt{makeglossaries.1}.

For comparison, consider the following document:
5.12 Sorting

\documentclass{article}
\usepackage[style={indexgroup}]{glossaries}
\makeglossaries
\newcommand*{\application}[1]{\texttt{#1}}
\newcommand*{\command}[1]{\texttt{\glsbackslash #1}}
\newglossaryentry{cs.makeglossaries}{%
  name={\command{makeglossaries}},
  description={opens glossary files}}
\newglossaryentry{ap.makeglossaries}{%
  name={\application{makeglossaries}},
  description={Perl script}}
\begin{document}
\gls{cs.makeglossaries} and \gls{ap.makeglossaries}.
\printglossaries
\end{document}

This uses makeindex, which puts both entries in the “Symbols” group (since they both start with \ from the start of \command and \application, respectively). The ordering is makeglossaries, \makeglossaries because “a” (second character of \application) comes before “c” (second character of \command).

The switch to xindy just involves adding the xindy package option:
\usepackage[xindy,style={indexgroup}]{glossaries}

This results in a glossary that only contains one entry, \makeglossaries, because xindy merges entries with duplicate sort values and the sort values end up as duplicates because xindy discards the \application and \command control sequences. Although bib2gls also ignores unknown control sequences, it doesn’t perform this merger.

If I add:
@preamble{"\providecommand*{\application}[1]{\texttt{#1}}\providecommand{\command}[1]{\texttt{\glsbackslash #1}}"}

to the earlier .bib file (called, say, entries.bib) then the document can be altered to use bib2gls:
\documentclass{article}
\usepackage[record,style={indexgroup}]{glossaries-extra}
5.12 Sorting

\GlsXtrLoadResources[\src={entries.bib},
sort-suffix={non-unique},
identical-sort-action={none}]

\begin{document}
\gls{cs.makeglossaries} and \gls{ap.makeglossaries}.
\printunsrtglossaries
\end{document}

This uses the default \texttt{sort={locale}} which considers an ignored (punctuation) character, so both \gls{makeglossaries} and \gls{makeglossaries} are listed in the “M” letter group, even though the interpreter has determined that the sort value for \texttt{cs.makeglossaries} is the literal string \texttt{makeglossaries}. Note that in this case \texttt{bib2gls} doesn’t detect duplicate sort values since it only uses a simple string comparison to detect duplicates rather than using the collator.

If I switch to using a letter-based sort rule instead, for example \texttt{sort={letter-nocase}}, then \texttt{makeglossaries} will be listed in the “Symbols” letter group since the leading \texttt{\} from the sort value \texttt{makeglossaries} isn’t ignored with this rule.

Now let’s suppose I use \texttt{interpret-preamble={false}} to prevent \texttt{bib2gls} from interpreting the preamble:

\GlsXtrLoadResources[\src={entries.bib},interpret-preamble={false}]

This means that the custom commands won’t be recognised and will therefore be ignored, so both entries will have their sort values reduced to \texttt{makeglossaries}.

The first entry to be processed is \texttt{cs.makeglossaries} because it’s the first to be selected. This is assigned the sort value \texttt{makeglossaries}. (Note that, unless you use \texttt{sort={unsrt}}, the initial selection order is based on the record order. In this example, \texttt{cs.makeglossaries} has the first record in the \texttt{.aux} file.)

The next entry to be processed is \texttt{ap.makeglossaries}. This also ends up with the sort value \texttt{makeglossaries} so \texttt{bib2gls} converts this to \texttt{makeglossaries1} and (with verbose mode on) the following messages are written to the transcript:

\begin{verbatim}
Sort value 'makeglossaries' (entry 'ap.makeglossaries') not unique for the entry's hierarchical level.
Appending suffix '1' to the sort value 'makeglossaries' for entry 'ap.makeglossaries'.
\end{verbatim}

Both entries are listed in the “M” letter group in the order \texttt{makeglossaries}, \texttt{makeglossaries}.

If the records are reversed:

\gls{ap.makeglossaries} and \gls{cs.makeglossaries}.

then the sort value for \texttt{cs.makeglossaries} is now considered the duplicate and the order is reversed: \texttt{makeglossaries}, \texttt{makeglossaries}.

Suppose now I modify the \texttt{.bib} file so that \texttt{ap.makeglossaries} is defined as:
5.12 Sorting

and suppose the document only contains an explicit reference to `ap.makeglossaries`:

\begin{document}
\gls{ap.makeglossaries}
\printunsrtglossaries
\end{document}

Now `ap.makeglossaries` is the first entry to be selected because entries with records are always selected before any (unrecorded) dependencies. In this case `cs.makeglossaries` is only selected because it’s required by `ap.makeglossaries`. Now `ap.makeglossaries` is the first to have its sort value assigned, and it’s `cs.makeglossaries` that has the duplicate. This means that the ordering in the glossary is now: `makeglossaries`, `\makeglossaries`.

An oddity occurs if the glossary is moved to the start of the document:

\begin{document}
\printunsrtglossaries
\gls{ap.makeglossaries}
\end{document}

In this case, the first document build:

\pdflatex myDoc
\bibgls --group --verbose myDoc
\pdflatex myDoc

leads to the ordering described above: `makeglossaries`, `\makeglossaries`. However, the next document build has a new record for `cs.makeglossaries` occurring in the glossary (within the description of `ap.makeglossaries`) which means it’s now the first entry to be selected so the ordering switches to: `\makeglossaries`, `makeglossaries`. In this type of situation you might be better off with the `identical-sort-action={id}` option instead.

Remember that you can temporarily switch off the indexing by locally setting:

\GlsXtrSetDefaultGlsOpts{noindex}

Since the glossary preamble is scoped, you can simply do

\appto\glossarypreamble{\GlsXtrSetDefaultGlsOpts{noindex}}

to switch off the indexing within the glossary (or use `\apptoglossarypreamble`). Note that this is different to using:

\GlsXtrSetDefaultNumberFormat{glsignore}

which creates an ignored record. Even though the record is ignored (and so won’t show in the location list) the record still influences the selection order and the record count.
5.12 Sorting

\texttt{sort-suffix-marker=\{value\}}

This automatically implements the dual and secondary settings \texttt{dual-sort-suffix-marker =\{\langle value\rangle\}} and \texttt{secondary-sort-suffix-marker=\{\langle value\rangle\}}.

If a suffix is appended to the sort value (see above) then it will be separated by the suffix marker, which can be set with \texttt{sort-suffix-marker=\{\langle value\rangle\}} where \langle value\rangle is the marker. By default the marker is empty. You can use \texttt{\string\u\langle hex\rangle} or \texttt{\gls\hex\langle hex\rangle} to indicate Unicode characters outside the ASCII range. If, for some reason, you want to use a special character, such as #, you will need to precede it with \texttt{\string} (for example \texttt{\string#}) or use the above hexadecimal markup. If you use \texttt{\#} it will be treated as a literal string containing a backslash followed by a hash character.

\texttt{encapsulate-sort=\{csname\}}

This option will encapsulate the sort value (after modifications such as \texttt{sort-suffix} and \texttt{sort-number-pad}) with \langle csname\rangle\{value\}\{entry-id\} where \langle value\rangle is the sort value that would otherwise have been used and \langle entry-id\rangle is the entry’s label. It will then be interpreted (if enabled). Note that \langle value\rangle may already have been interpreted in a previous step.

\texttt{strength=\{value\}}

This option automatically implements \texttt{dual-strength=\{\langle value\rangle\}} and \texttt{secondary-strength=\{\langle value\rangle\}}.

The collation strength used by the alphabet sort methods (table 5.2) can be set to the following values: primary (default), secondary, tertiary or identical. These indicate the difference between two characters, but the exact assignment is locale dependent. See the documentation for Java’s \texttt{Collator} class [3] for further details.

For example, suppose the file \texttt{entries.bib} contains:

@index{resume}
@index{RESUME}
@index{resumee, name={r\'esum\'e}}
@index{rat}
@index{rot}
@index{aardvark}
@index{zoo}

and the document contains:

\documentclass{article}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[sort={en},src={entries}]
5.12 Sorting

\begin{document}
\gls{resumee}, \gls{resume}, \gls{RESUME}, \\
\gls{aardvark}, \gls{rat}, \gls{rot}, \gls{zoo}.
\printunsrtglossaries
\end{document}

then this uses the default \texttt{strength=primary}, so the entries are listed as aardvark, rat, résumé, resume, RESUME, rot, zoo.

If the strength is changed to secondary:

\GlsXtrLoadResources[sort={en},src={entries},strength={secondary}]

then the entries are listed as aardvark, rat, resume, RESUME, résumé, rot, zoo.

If the strength is changed to tertiary or identical, there's no difference from \texttt{strength=secondary} for this particular example.

This option is ignored by non-alphabet sorts (such as letter or numeric).

decomposition=\langle value \rangle

This option automatically implements the dual and secondary settings \texttt{dual-decomposition=}\langle value \rangle and \texttt{secondary-decomposition=}\langle value \rangle.

The collation decomposition used by alphabet sort methods (table 5.2) can be set to the following values: \texttt{canonical} (default), \texttt{full} or \texttt{none}. This determines how Unicode composed characters are handled. The fastest mode is \texttt{none} but is only appropriate for languages without accents. The slowest mode is \texttt{full} but is the most complete for languages with non-ASCII characters. See the documentation for Java's \texttt{Collator} class [3] for further details. This option is ignored by non-alphabet sorts (such as letter or numeric).

letter-number-rule=\langle value \rangle

This automatically implements the dual and secondary settings \texttt{dual-letter-number-rule=}\langle value \rangle and \texttt{secondary-letter-number-rule=}\langle value \rangle.

If you use one of the letter-number sort methods (table 5.4), then you can determine the comparison between a number and letter. The \langle value \rangle may be one of:

- \texttt{before letter}: numbers are considered less than any letter.
- \texttt{after letter}: numbers are considered greater than any letter.
- \texttt{between}: (default) numbers come between letter cases. With the \texttt{letternumber-case} sort option, this will put numbers after upper case and before lower case. This setting doesn’t make much sense with the \texttt{letternumber-nocase} option but, if used, this will put numbers before letters. The \texttt{letternumber-upperlower} and \texttt{letternumber-lowerupper} options are more complicated. See section 5.12 for more detail.

289
5.12 Sorting

- **first**: numbers are considered less than all characters (including punctuation and spaces).

- **last**: numbers are considered greater than all characters (including punctuation and spaces).

Note that the reverse sort methods will invert this setting. Remember also that the case-insensitive letter-number sort methods always first convert the `sort` field to lower case, which means that if you use one of them then there won’t be any upper case characters.

Use `letter-number-punc-rule` to determine the relative position of white space and punctuation.

**letter-number-punc-rule=⟨value⟩**

This automatically implements the dual and secondary `dual-letter-number-punc-rule=⟨(value)⟩` and `secondary-letter-number-punc-rule=⟨(value)⟩`.

If you use one of the letter-number sort methods (table 5.4), then you can determine the order of white space and punctuation. In this context, punctuation means any character that’s not considered a letter, a number or white space. This means that characters such as combining marks are considered punctuation.

The `⟨value⟩` may be one of the following:

- **punc-space-first**: punctuation comes first, followed by white space (then letters and optionally numbers according to the letter-number rule);

- **punc-space-last**: punctuation followed by white space come last (after letters and optionally numbers according to the letter-number rule);

- **space-punc-first**: white space comes first, followed by punctuation (then letters and optionally numbers according to the letter-number rule);

- **space-punc-last**: white space followed by punctuation come last (after letters and optionally numbers according to the letter-number rule);

- **space-first-punc-last**: white space comes first (followed by letters and optionally numbers according to the letter-number rule) and punctuation comes last;

- **punc-first-space-last**: punctuation comes first (followed by letters and optionally numbers according to the letter-number rule) and white space comes last;

- **punc-first-space-zero**: punctuation comes first (although numbers may come before) and white space is replaced by the digit 0 (0x30);

- **punc-last-space-zero**: punctuation comes last (although numbers may come after) and white space is replaced by the digit 0 (0x30).
5.12 Sorting

- punc-first-space-zero-match-next: punctuation comes first (although numbers may come before) and white space is replaced by the appropriate zero character (see below);

- punc-last-space-zero-match-next: punctuation comes last (although numbers may come after) and white space is replaced by the appropriate zero character (see below).

Remember that the reverse sort methods will invert order governed by this setting.

For the space-zero-match-next settings, the sort value will have all spaces replaced with a digit that represents zero. If the space isn’t followed by a digit, the basic Latin 0 (0x30) will be used, otherwise bib2gls will try to match the zero with the following digit group. For example, if the space is followed by 1 (0xB9) the space will be replaced by 0 (0x2070), resulting in the sub-string 01 (0xB9 0x2070).

If just the space-zero (without the -match-next) is used then the space will just be replaced with 0 resulting in the sub-string 01 (0x30 0x2070). In this case, the 0 will be distinct from 1 (rather than being considered a leading zero). However, for other numbering systems the 0 will be treated as a leading zero. For example, if the space is followed by the Devanagari digit one (0x0967) then the sub-string will be 0x30 0x0967 but here the mixture is allowed to form a number (with a leading zero) as both characters belong to the Unicode category “Number, Decimal Digit”.

This means that the -match-next settings are only really needed if the sort string contains the superscript or subscript digits that don’t belong to the “Number, Decimal Digit” category. The plain space-zero alternatives are more efficient as they just perform a simple substitution.

The \TeX parser library used by bib2gls recognises the standard \LaTeX text-mode commands \textsuperscript{⟨text⟩} and \textsubscript{⟨text⟩} and will use the Unicode superscript or subscript characters if they cover every character in ⟨text⟩, otherwise HTML markup is used, but that’s then stripped by bib2gls. This means that:

C\textsubscript{10}H\textsubscript{10}O\textsubscript{4}

will be converted to: C_{10}H_{10}O_{4} but:

X\textsubscript{1, 2}

will be converted to:

X_{1, 2}

which ends up as X_{1, 2}.

Note that letter-number-rule={first} and letter-number-rule={last} overrides this option when comparing a number with white space or punctuation.
5.12 Sorting

**numeric-sort-pattern={value}**

If you use the custom `sort={numberformat}` or `sort={numberformat-reverse}`, you need to specify the format pattern with this option where `{value}` is a pattern recognised by Java’s `java.text.DecimalFormat` class [4]. You can use `\string\u{hex}` or `\gl\hex{hex}` to indicate Unicode characters by their hexadecimal code. You can also use `\#`, `\%`, `\_`, `\&`, `{ and `}` to indicate `#`, `%`, `_`, `&`, `{ and `}`.

Where the dual or secondary sort uses `numberformat` or `numberformat-reverse`, use `dual-numeric-sort-pattern` for `dual-sort` and `secondary-numeric-sort-pattern` for `secondary`.

**numeric-locale={value}**

If you use any of the locale-sensitive numeric sort methods described in section 5.12, such as `sort={numeric}`, use this option to set the locale if the default resource locale isn’t appropriate. The value may be:

- **resource**: use the default resource locale, if set, otherwise assume `doc`;
- **doc**: use the document locale or, if not set, assume `numeric-locale={locale};`
- **locale**: use the Java locale (which is usually the operating system’s locale);
- `{lang-tag}`: set to the locale identified by the given a valid language tag `{lang-tag}`.

Use `dual-numeric-locale` for `dual-sort` and `secondary-numeric-locale` for `secondary`.

If you use the `locale` resource option with `numeric-locale={resource}`, then the `locale` option must be come before `numeric-locale`.

**date-sort-locale={value}**

If you use a date/time sort method (table 5.6), then you can set the locale used by Java’s date-time parser. The default setting is `date-sort-locale={resource}`.

The value may be `resource` (use the resource locale), `doc` (use the document locale), `locale` (use the Java locale), or a valid language tag `{lang-tag}` identifying the locale.

Use `dual-date-sort-locale` and `secondary-date-sort-locale` for the dual and secondary.

If you use the `locale` resource option with `date-sort-locale={resource}`, then the `locale` option must be come before `date-sort-locale`.
5.12 Sorting

date-sort-format=⟨value⟩

If you use a date/time sort method (table 5.6), then you can set the format used by Java’s date-time parser. If omitted, date-sort-format={default} is assumed. The ⟨value⟩ may be one of:

- default: use the locale’s default format.
- short: use the locale’s short format.
- medium: use the locale’s medium format.
- long: use the locale’s long format.
- full: use the locale’s full format.

- ⟨pattern⟩: provide a custom pattern. This should match the specifications for Java’s SimpleDateFormat class [7]. You may use \string\u⟨hex⟩ or \glshex ⟨hex⟩ to indicate Unicode characters or \#, \%, \_, \& \{ and \} to indicate #, %, _, & { and }.

With the custom setting, if the pattern only contains date (but not time) information, then it must be used with sort={date} or sort={date-reverse}. If the pattern only contains time (but not date) information, then it must be used with sort={time} or sort={time-reverse}. If the pattern contains date and time information, then it must be used with sort={datetime} or sort={datetime-reverse}.

For example, suppose each entry provides information about a person and the user1 field is used to store their date of birth:

@entry{caesar,
  name={Gaius Julius Caesar},
  first={Julius Caesar},
  text={Caesar},
  description={Roman politicians and general},
  user1={13 July 100 BC}
}

@entry{wellington,
  name={Arthur Wellesley, 1st Duke of Wellington},
  first={Arthur Wellesley (Duke of Wellington)},
  text={Wellington},
  description={Anglo-Irish soldier and statesman},
  user1={1 May 1769 AD}
}

Then the entries can be sorted by date of birth using:
5.12 Sorting

\GlsXtrLoadResources[
  src={entries},% data in entries.bib
  sort-field={user1},
  sort={date},
  date-sort-format={d MMM y G}
]

The G (era) date pattern specifier expects a string, such as “AD”. It will match lower case forms, such as “ad”, so if you have \textsc{ad} the interpreter will convert this to ad (stripping the text-block command). However, in general it’s best to supply a semantic command that ensures that the interpreted result matches the required format.

For example, if \era is provided with:

@preamble{"\providecommand{\era}[1]{\textsc{\MakeLowercase{#1}}}"}

If the definition is hidden from the interpreter (interpret-preamble={false}) and the field value contains \era{AD} then the custom command will simply be stripped leaving AD which can be matched by G.

If the definition is picked up by the interpreter then the field value will contain ad (from \MakeLowercase) but this can be matched by G, so it isn’t a problem. However, if the definition of \era is changed so that the era label supplied in the argument is converted to something that doesn’t match G then the definition should be hidden from the interpreter.

Here’s a complete document that changes the group fields to use the year and era:

\documentclass{article}
\usepackage[record,style={indexgroup}]{glossaries-extra}
\newcommand{\bibglstdategroup}[7]{#1#4#7}
\newcommand{\bibglstdategrouptitle}[7]{\number#1\␣#4}
\GlsXtrLoadResources[
  src={entries},
  sort-field={user1},
  sort={date},
  date-sort-format={d MMM y G},
  selection={all}
]
\begin{document}
\printunsrtglossaries
\end{document}

(The use of \number strips the leading zero from the year.)
5.13 Secondary Glossary

**group-formation**=\{value\}

If the group field hasn’t been set in the .bib file or through options like group, then it is assigned according to this option’s setting during sorting if --group has been used. Permitted values:

- **default**: the group is assigned according to the sort method’s default group formation. This is the default setting.
- **codepoint**: the group is set to \bibglsunicodegroup{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}, where the first argument is the first significant character (converted to lower case and decomposed, if applicable) of the sort value.
- **unicode category**: the group is set to \bibglsunicodegroup{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}, where the first argument is the label identifying the Unicode category of the first significant character of the sort value. For example, the label Ll signifies a lower case letter and Lu signifies an upper case letter.
- **unicode script**: the group is set to \bibglsunicodegroup{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}, where the first argument is the label identifying the Unicode script of the first significant character of the sort value. For example, the label LATIN indicates Latin, GREEK indicates Greek and COMMON indicates common characters (such as mathematical Greek characters that are often used with non-Greek scripts).
- **unicode category and script**: the group is set to \bibglsunicodegroup{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}, where the first argument is the label corresponding to the Unicode category and script of the first significant character of the sort value. For example, the label Ll.LATIN indicates a lower case Latin letter.

This option has no effect with --no-group or if no sorting is applied. Use secondary-group-formation for secondary sorting and dual-group-formation for dual entries.

Settings other than the default can cause the groups to become fragmented, so care is needed if you use this option. See also section 1.2.

5.13 Secondary Glossary

The secondary glossary may only be used with action={define} (within the same resource set) since it’s incompatible with the copy actions. You may use secondary in the first resource set and a copy action in a subsequent resource set.

**secondary**=\{value\}

It may be that you want to display a glossary twice but with a different order. For example, the first time alphabetically and the second time by category. One way to do this is to have
two \GlsXtrLoadResources that both load the same .bib file with different label-prefix and sort settings, but this is only possible with selection={all} or by ensuring you reference each entry with both label prefixes. Another method is to use action={copy} but this requires a second resource command with the same selection criteria.

A simpler method is to use a single \GlsXtrLoadResources with the secondary option. The value (which must be supplied) should be in the format:

\begin{verbatim}
⟨sort⟩:⟨field⟩:⟨type⟩
\end{verbatim}
or
\begin{verbatim}
⟨sort⟩:⟨type⟩
\end{verbatim}

If the ⟨field⟩ is omitted, the value of sort-field is used. Remember that when the primary entries are sorted, the sort field will be set, which means that the sort fallback field (see section 5.8) won’t be used in the secondary sort. In general it’s best to supply the field unless one type is sorted and the other isn’t. (The actual sort value obtained by the secondary sort will be saved in the secondary sort field in case you require it.)

The value of ⟨sort⟩ is as for sort, but note that in this case the sort value unsrt or none means to use the same ordering as the primary entries. For example, with sort={de-CH-1996}, secondary={none:copies} the copies list will be ordered according to de-CH-1996 and not according to the order in which they were read when the .bib file or files were parsed. If ⟨sort⟩ is custom, then the rule should be provided with secondary-sort-rule.

This option will copy all the selected entries into the glossary labelled ⟨type⟩ sorted according to ⟨sort⟩ (using ⟨field⟩ as the sort value). Note that this just copies the entry’s label to the secondary glossary list rather than creating a duplicate entry, which saves resources but it means that all the fields will be identical. If you want groups in your glossary, the group information for the secondary glossaries will be stored in the internal secondary group field. The group field will contain the group for the primary glossary.

In order to switch fields in \printunsrtglossary, you need at least v1.21 of glossaries-extra which provides \glsxtrgroupfield to keep track of the appropriate field label. If this command is defined, the preamble for the secondary glossary will be adjusted to locally change the field to secondary group. With older versions, the group information in the secondary glossary will be the same as for the primary glossary.

If the glossary ⟨type⟩ doesn’t exist, it will be defined with \provideignoredglossary* \{⟨type⟩\} even if --no-provide-glossaries is set. Note that if the glossary already exists and contains entries, the existing entries aren’t re-ordered. The new entries are simply appended to the list.

For example, suppose the .bib file contains entries like:

\begin{verbatim}
@entry{quartz,
    name={quartz},
    description={hard mineral consisting of silica},
    category={mineral}
}
\end{verbatim}
5.13 Secondary Glossary

@entry{cabbage,
   name={cabbage},
   description={vegetable with thick green or purple leaves},
   category={vegetable}
}

@entry{waterfowl,
   name={waterfowl},
   description={any bird that lives in or about water},
   category={animal}
}

and the document preamble contains:

\GlsXtrLoadResources[src={entries},sort={en-GB},
   secondary={en-GB:category:topic}
]

This sorts the primary entries according to the default sort-field and then sorts the entries according to the category field and copies this list to the topic glossary (which will be provided if not defined.)

The secondary list can be displayed with the hypertargets switched off to prevent duplicates. The cross-references will link to the original glossary.

For example:

\printunsrtglossary[title={Summary (alphabetical)}]
\printunsrtglossary[title={Summary (by topic)},target={false}]

The alternative (or if more than two lists are required) is to reload the same .bib file with different label prefixes. For example, if the entries are stored in entries.bib:

\newglossary*{nosort}{Symbols (Unsorted)}
\newglossary*{byname}{Symbols (Letter Order)}
\newglossary*{bydesc}{Symbols (Ordered by Description)}
\newglossary*{byid}{Symbols (Ordered by Label)}

\GlsXtrLoadResources[
   src={entries},% entries.bib
   sort={unsrt},
   type={nosort}
]

\GlsXtrLoadResources[
   src={entries},% entries.bib
   sort={letter-case},
   type={byname},
]
5.13 Secondary Glossary

\GlsXtrLoadResources[
  src={entries},% entries.bib
  sort={locale},
  sort-field={description},
  type={bydesc},
  label-prefix={bydesc.}
]

\GlsXtrLoadResources[
  src={entries},% entries.bib
  sort={letter},
  sort-field={id},
  type={byid},
  label-prefix={byid.}
]

**secondary-match**=\langle key=value list \rangle

Similar to `match` but determines whether or not to include a primary entry in the secondary list. The syntax is the same but this option is governed by `secondary-match-op` and `secondary-match-action`. Note that if an entry hasn’t been selected for the primary list then it won’t be added to the secondary list, regardless of this setting.

**secondary-not-match**=\langle key=value list \rangle

Similar to `not-match` but determines whether or not to include a primary entry in the secondary list. The syntax is the same but this option is governed by `secondary-match-op` and `secondary-match-action`. Note that if an entry hasn’t been selected for the primary list then it won’t be added to the secondary list, regardless of this setting.

**secondary-match-op**=\langle value \rangle

As `match-op` but for the secondary list selection.

**secondary-match-action**=\langle value \rangle

As `match-action` but for the secondary list selection.

**secondary-missing-sort-fallback**=\langle field \rangle

As `missing-sort-fallback` but for secondary sorting.
secondary-trim-sort=⟨boolean⟩
As trim-sort but for secondary sorting.

secondary-sort-replace=⟨list⟩
As sort-replace but for secondary sorting.

secondary-sort-rule=⟨value⟩
As sort-rule but for secondary custom sorting.

secondary-break-at=⟨value⟩
As break-at but for secondary entries.

secondary-break-marker=⟨marker⟩
As break-marker but for secondary entries.

secondary-break-at-match=⟨key=value list⟩
As break-at-match but for secondary entries.

secondary-break-at-match-op=⟨value⟩
As break-at-match-op but for secondary entries.

secondary-break-at-not-match=⟨key=value list⟩
As break-at-not-match but for secondary entries.

secondary-sort-number-pad=⟨number⟩
As sort-number-pad but for secondary entries.

secondary-sort-pad-plus=⟨marker⟩
As sort-pad-plus but for secondary entries.

secondary-sort-pad-minus=⟨marker⟩
As sort-pad-minus but for secondary entries.
secondary-identical-sort-action=⟨value⟩
As identical-sort-action but for secondary entries.

secondary-sort-suffix=⟨value⟩
As sort-suffix but for secondary entries.

secondary-sort-suffix-marker=⟨value⟩
As sort-suffix-marker but for secondary entries.

secondary-strength=⟨value⟩
As strength but for secondary entries.

secondary-decomposition=⟨value⟩
As decomposition but for secondary entries.

secondary-letter-number-rule=⟨value⟩
As letter-number-rule but for secondary letter-number sorting.

secondary-letter-number-punc-rule=⟨value⟩
As letter-number-punc-rule but for secondary letter-number sorting.

secondary-numeric-sort-pattern=⟨value⟩
As numeric-sort-pattern but for secondary locale-sensitive numeric sorting.

secondary-numeric-locale=⟨value⟩
As numeric-locale but for secondary locale-sensitive numeric sorting.

secondary-date-sort-locale=⟨value⟩
As date-sort-locale but for secondary date-time sorting.

secondary-date-sort-format=⟨value⟩
As date-sort-format but for secondary date-time sorting.
secondary-group-formation=\langle value \rangle

As group-formation but for secondary sorting.

5.14 Dual Entries

General Dual Settings

dual-prefix=\langle value \rangle

This option indicates the prefix to use for the dual entries. The default value is dual. (including the terminating period). Any references to dual entries within the .bib file should use the prefix dual. which will be replaced by \langle value \rangle when the .bib file is parsed.

As from version 1.8, the dual label prefix is identified in the .gls\text{tex} file with:

\bibglsdualprefixlabel\{\langle prefix \rangle\}

primary-dual-dependency=\langle boolean \rangle

This is a boolean setting that determines whether or not primary and dual entries should be considered mutual dependencies. The default value is primary-dual-dependency=\{true\}, which means that if a primary has records then the dual is added as a dependency and vice versa. The setting primary-dual-dependency=\{false\} can’t be used with dual-sort=\{none\} or dual-sort=\{use\} (but may be used with dual-sort=\{combine\} and sort=\{none\} or sort=\{use\}).

combine-dual-locations=\langle value \rangle

This setting allows the location lists for each primary entry to be merged with that of the corresponding dual entry. The \langle value \rangle may be one of:

- false This is the default setting. The location lists aren’t combined.
- both Both the primary and dual are given the combined location list.
- dual Only the dual is given the combined location list. The primary’s location list is emptied.
- primary Only the primary is given the combined location list. The dual’s location list is emptied.
- dual retain principal Like dual but any principal locations for primary entries will have a copy left in the primary entry’s location list.
- primary retain principal Like primary but any principal locations for dual entries will have a copy left in the dual entry’s location list.

For example, suppose the file entries.bib contains:
5.14 Dual Entries

@dualindexentry{array,
   description={ordered list of values}
}

@dualindexentry{vector,
   name={vector},
   description={column or row of values}
}

@dualindexentry{set,
   description={collection of values}
}

@dualindexentry{matrix,
   plural={matrices},
   description={rectangular array of values}
}

and the document contains:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,index,style={indexgroup}]{glossaries-extra}
\GlsXtrLoadResources[
   src={entries},
   type={index},
   label-prefix={idx.},
   dual-prefix={gls.},
   dual-type={main}
]
\begin{document}
\gls{gls.array}, \gls{gls.vector}, \gls{gls.set}, \gls{gls.matrix}.
\newpage
\gls{gls.array}, \gls{idx.vector}, \gls{idx.set}, \gls{gls.matrix}.
\newpage
\gls{gls.array}, \gls{gls.vector}, \gls{gls.set}, \gls{gls.matrix}.
\printunsrtglossaries
\end{document}
In this case, the primary entries are placed in the index glossary type and are assigned the prefix idx. but only two of the primary entries have been used in the document (both on page 2).

The dual entries are assigned the prefix gls. and are placed in the main glossary. The gls.array and gls.matrix entries have been indexed on pages 1, 2 and 3. The gls.vector and gls.set entries have been indexed on pages 1 and 3.

With the default setting, some of the locations are in the main glossary (corresponding to \gls{gls.array}, \gls{gls.vector}, \gls{gls.set} and \gls{gls.matrix}) and some of the locations are in the index glossary (corresponding to \gls{idx.vector} and \gls{idx.set}).

If the option combine-dual-locations={primary} is added to the resource set, then all the locations are moved to the index glossary. The entries in the main glossary no longer have locations. This is actually preferable for this type of document and it’s best not to reference the primary (index) entries as the hyperlink created by \gls will point to the index, but these entries don’t have descriptions, so it’s less useful than referencing the dual (main) entries as then the hyperlink can point to the definition in the main glossary.

**Dual Fields**

dual-type={value}

This option sets the type field for all dual entries. (The primary entries obey the type option.) This will override any value of type provided in the .bib file (or created through a mapping). The {value} is required and should be one of:

- **false**: switches off this setting (default);
- **same as entry**: sets the type to the entry type (lower case and without the initial @). For example, if the entry was defined with @dualentry, the type will be set to dualentry. If you’ve used entry-type-aliases, this refers to the target entry type not the original entry type provided in the .bib file.
- **same as original entry**: set the type field to the original entry type (lower case and without the initial @) before it was aliased (behaves like same as entry if the entry type wasn’t aliased).
- **same as base**: sets the type to the base name of the .bib file (without the extension) that provided the entry definition (new to v1.1);
- **same as primary**: sets the type to the same as the corresponding primary entry’s type (which may have been set with type). If the primary entry doesn’t have the type field set, the dual’s type will remain unchanged.
- **same as parent**: sets the type to the same as the entry’s parent (new to v1.9). If the entry doesn’t have a parent or if the parent doesn’t have the type field set, then no change is made.
5.14 Dual Entries

- same as category set the type field to the same value as the category field (type unchanged if category not set);

- \langle label \rangle: sets the type field to \langle label \rangle.

Remember that the glossary with that label must have already been defined (see section 1.3). For example:

\begin{verbatim}
\newglossary*{english}{English}
\newglossary*{french}{French}
\end{verbatim}

\begin{verbatim}
\GlsXtrLoadResources[src={entries},sort={en},dual-sort={fr},
    type={english},
    dual-type={french}]
\end{verbatim}

Alternatively:

\begin{verbatim}
\newglossary*{dictionary}{Dictionary}
\end{verbatim}

\begin{verbatim}
\GlsXtrLoadResources[src={entries},sort={en},dual-sort={fr},
    type={dictionary},
    dual-type={same as primary}]
\end{verbatim}

dual-category=\langle value \rangle

This option sets the category field for all dual entries. (The primary entries obey the category option.) This will override any value of category provided in the .bib file (or created through a mapping). The \langle value \rangle may be empty or one of:

- false: switch off this setting (default);

- same as entry: sets the category to the entry type (lower case and without
  the initial @). For example, if the entry was defined with \texttt{@dualentry}, the category will
  be set to dualentry. If you’ve used \texttt{entry-type-aliases}, this refers to the target
  entry type not the original entry type provided in the .bib file.

- same as original entry: set the category field to the original entry type (lower
  case and without the initial @) before it was aliased (behaves like same as entry if
  the entry type wasn’t aliased).

- same as base: sets the category to the base name of the .bib file (without the
  extension) that provided the entry definition (new to v1.1);

- same as primary: sets the category to the same as the corresponding primary en-
  try’s category (which may have been set with category). If the primary entry doesn’t
  have the category field set, the dual’s category will remain unchanged.
5.14 Dual Entries

- **same as type**: sets the category to the same as the value of the entry’s type field (which may have been set with dual-type). If the entry doesn’t have the type field set, the category will remain unchanged.

- ⟨label⟩: sets the category field to ⟨label⟩.

```
dual-counter=⟨value⟩
```

As counter but for the dual entries. In this case ⟨value⟩ may be the name of the counter or same as primary which uses the counter for the primary entry or false to switch off this setting.

```
dual-short-case-change=⟨value⟩
```

As short-case-change but applies to the dualshort field instead.

```
dual-long-case-change=⟨value⟩
```

As long-case-change but applies to the duallong field instead.

```
dual-field=⟨value⟩
```

If this option is used, this will add \glsxtrprovidestoragekey to the start of the .glstex file providing the key given by ⟨value⟩. Any entries defined using a dual entry type, such as @dualentry, will be written to the .glstex file with an extra field called ⟨value⟩ that is set to the mirror entry. If ⟨value⟩ is omitted dual-field={dual} is assumed. If you use a different value, you will need to redefine \GlsXtrDualField (either locally or globally). A value of false will switch off this setting (the default).

For example, if the .bib file contains:

```
@dualentry{child,  
  name={child},  
  plural={children},  
  description={enfant} 
}
```

Then with dual-field={dual} (or simply dual-field without a value) this will first add the line:

```
\glsxtrprovidestoragekey{dual}{}{}
```

at the start of the file and will include the line:

```
dual={dual.child},
```

for the primary entry (child) and the line:

```
dual={child},
```
for the dual entry (dual.child). It’s then possible to reference one entry from the other. For example, the post-description hook could contain:

\ifglsishasfield{dual}\{\glscurrententrylabel\}
  \%  
  \space  
  (\glshyperlink{\glscurrentfieldvalue})\%  
\}%
%

Note that this new field won’t be available for use within the .bib file (unless it was previously defined in the document before \glsxtrresourcefile).

dual-date-time-field-format=(value)
As date-time-field-format but is used for dual entries.

dual-date-field-format=(value)
As date-field-format but is used for dual entries.

dual-time-field-format=(value)
As time-field-format but is used for dual entries.

dual-date-time-field-locale=(value)
As date-time-field-locale but is used for dual entries.

dual-date-field-locale=(value)
As date-field-locale but is used for dual entries.

date-time-field-locale=(value)
As time-field-locale but is used for dual entries.

**Dual Sorting**

dual-sort=(value)
This option indicates how to sort the dual entries. The primary entries are sorted with the normal entries according to sort, and the dual entries are sorted according to dual-sort unless dual-sort={combine} in which case the dual entries will be combined with the primary entries and all the entries will be sorted together according to the sort option.

If \langle value \rangle isn’t set to combine then the dual entries are sorted separately according to \langle value \rangle (as per sort) and the dual entries will be appended at the end of the .glistex file.
The field used by the comparator is given by `dual-sort-field`. If `dual-sort={custom}`, then the dual entries are sorted according to the rule provided by `dual-sort-rule`.

For example:

```
\GlsXtrLoadResources[
    src={entries-dual},
    sort={en},
    dual-sort={de-CH-1996}
]
```

This will sort the primary entries according to `en` (English) and the secondary entries according to `de-CH-1996` (Swiss German new orthography) whereas:

```
\GlsXtrLoadResources[
    src={entries-dual},
    sort={en-GB},
    dual-sort={combine}
]
```

will combine the dual entries with the primary entries and sort them all according to the `en-GB` locale (British English).

If not set, `dual-sort` defaults to `combine`. If `<value>` is omitted, `resource` is assumed.

`dual-sort-field={(field)}`

This option indicates the field to use when sorting dual entries (when they haven’t been combined with the primary entries). The default value is the same as the `sort-field` value.

`dual-missing-sort-fallback={(field)}`

As `missing-sort-fallback` but for dual sorting.

`dual-trim-sort={(boolean)}`

As `trim-sort` but for dual sorting.

`dual-sort-replace={(list)}`

As `sort-replace` but for dual sorting.

`dual-sort-rule={(value)}`

As `sort-rule` but for `dual-sort={custom}`.

`dual-break-at={(value)}`

As `break-at` but for dual entries.
5.14 Dual Entries

dual-break-marker=⟨marker⟩
As break-marker but for dual entries.

dual-break-at-match=⟨key=value list⟩
As break-at-match but for dual entries.

dual-break-at-match-op=⟨value⟩
As break-at-match-op but for dual entries.

dual-break-at-not-match=⟨key=value list⟩
As break-at-not-match but for dual entries.

dual-sort-number-pad=⟨number⟩
As sort-number-pad but for dual entries.

dual-sort-pad-plus=⟨marker⟩
As sort-pad-plus but for dual entries.

dual-sort-pad-minus=⟨marker⟩
As sort-pad-minus but for dual entries.

dual-identical-sort-action=⟨value⟩
As identical-sort-action but for dual entries.

dual-sort-suffix=⟨value⟩
As sort-suffix but for dual entries.

dual-sort-suffix-marker=⟨value⟩
As sort-suffix-marker but for dual entries.

dual-strength=⟨value⟩
As strength but for dual entries.

dual-decomposition=⟨value⟩
As decomposition but for dual entries.
5.14 Dual Entries

dual-letter-number-rule=⟨value⟩
As letter-number-rule but for dual entries that use a letter-number sort.

dual-letter-number-punc-rule=⟨value⟩
As letter-number-punc-rule but for dual entries that use a letter-number sort.

dual-numeric-sort-pattern=⟨value⟩
As numeric-sort-pattern but for dual entries that use a locale-sensitive numeric sort.

dual-numeric-locale=⟨value⟩
As numeric-locale but for dual entries that use a locale-sensitive numeric sort.

dual-date-sort-locale=⟨value⟩
As date-sort-locale but for dual entries that use a date/time sort.

dual-date-sort-format=⟨value⟩
As date-sort-format but for dual entries that use a date/time sort.

dual-group-formation=⟨value⟩
As group-formation but for dual sorting.

Dual Mappings

dual-entry-map=ṃ{{⟨list1⟩},⟨⟨list2⟩⟩}

This setting governs the behaviour of @dualentry definitions. The value consists of two comma-separated lists of equal length identifying the field mapping used to create the dual entry from the primary one. Note that the alias field can’t be mapped.

The default setting is:

dual-entry-map={
   {name,plural,description,descriptionplural},
   {description,descriptionplural,name,plural}
}

The dual entry is created by copying the value of the field in the first list ⟨list1⟩ to the field in the corresponding place in the second list ⟨list2⟩. Any additional fields are copied over to the same field. For example:
5.14 Dual Entries

@dualentry{cat,
    name={cat},
    description={chat},
    see={dog}
}

defines two entries. The primary entry is essentially like:

@entry{cat,
    name={cat},
    plural={cat\glspluralsuffix },
    description={chat},
    descriptionplural={chat\glspluralsuffix },
    see={dog}
}

and the dual entry is essentially like:

@entry{dual.cat,
    description={cat},
    descriptionplural={cat\glspluralsuffix },
    name={chat},
    plural={chat\glspluralsuffix },
    see={dog}
}

(except they’re defined using \bibglsnewdualentry instead of \bibglsnewentry, and each is considered dependent on the other.)

The see field isn’t listed in dual-entry-map so its value is simply copied directly over to the see field in the dual entry. Note that the missing plural and descriptionplural fields have been filled in using their fallback values (see section 5.8).

In general bib2gls doesn’t try to supply missing fields, but in the dual entry cases it needs to do this for the mapped fields. This is because the shuffled fields might have different default values from the glossaries-extra package’s point of view. For example, \longnewglossary-entry doesn’t provide a default for descriptionplural if it hasn’t been set.

dual-abbrv-map=\{(\{list1\}\},\{(\{list2\}\})

This is like dual-entry-map but applies to @dualabbreviation rather than @dualentry. Note that the alias field can’t be mapped. The default setting is:

dual-abbrv-map=
    \{short,shortplural,long,longplural,dualshort,dualshortplural,
        duallong,duallongplural\},
    \{dualshort,dualshortplural,duallong,duallongplural,short,shortplural,
        long,longplural\}
5.14 Dual Entries

This essentially flips the short field with the dualshort field and the long field with the duallong field. See @dualabbreviation for further details.

dual-abbrventry-map={{list1},{list2}}

This is like dual-entry-map but applies to @dualabbreviationentry rather than @dualentry. Note that the alias field can’t be mapped. The default setting is:

dual-abbrventry-map=
  {long,short,shortplural},
  {name,text,plural}
}

See @dualabbreviationentry for further details.

dual-symbol-map={{list1},{list2}}

This is like dual-entry-map but applies to @dualsymbol rather than @dualentry. Note that the alias field can’t be mapped. The default setting is:

dual-symbol-map=
  {name,plural,symbol,symbolplural},
  {symbol,symbolplural,name,plural}
}

This essentially flips the name field with the symbol field.

dual-indexentry-map={{list1},{list2}}

This is like dual-entry-map but applies to @dualindexentry rather than @dualentry. The default setting is:

dual-indexentry-map=
  {name},
  {name}
}

Note that there must always be at least one pair, even if it’s the same field, since this identifies the field to use for the backlink, if set.

dual-indexsymbol-map={{list1},{list2}}

This is like dual-entry-map but applies to both @dualindexsymbol and @dualindexnumber. The default setting is:

dual-indexsymbol-map=
  {symbol,name,symbolplural,plural},
  {name,symbol,plural,symbolplural}
}
5.14 Dual Entries

dual-indexabbrv-map={{\text{list1}},{{\text{list2}}}}

This is like dual-entry-map but applies to both the dual \texttt{@dualindexabbreviation} and tertiary \texttt{@tertiaryindexabbreviationentry} entry types. The default setting is:

dual-indexabbrv-map={
    {name},
    {name}
}

**Dual Back-Links**

dual-entry-backlink={\langle boolean\rangle}

This is a boolean setting. If \langle boolean\rangle is missing \texttt{true} is assumed.

When used with \texttt{@dualentry}, if \langle boolean\rangle is \texttt{true}, this will wrap the contents of the first mapped field with \texttt{\bibglshyperlink}. The field is obtained from the first mapping listed in dual-entry-map.

For example, if the document contains:

\begin{verbatim}
\GlsXtrLoadResources[dual-entry-backlink,
    dual-entry-map={
        {name,plural,description,descriptionplural},
        {description,descriptionplural,name,plural}
    },
    src={entries-dual}]
\end{verbatim}

and if the .bib file contains:

\begin{verbatim}
@dualentry{child,
    name={child},
    plural={children},
    description={enfant}
}
\end{verbatim}

Then the definition of the primary entry (child) in the .glstex file will set the description field to:

\begin{verbatim}
\bibglshyperlink{enfant}{dual.child}
\end{verbatim}

and the dual entry (dual.child) will have the description field set to:

\begin{verbatim}
\bibglshyperlink{child}{child}
\end{verbatim}

This use of the wrapper \texttt{\bibglshyperlink} (rather than explicitly using \texttt{\glshyperlink}) and inserting the actual field value (rather than using commands like \texttt{\glsentryname}) allows it to work with \texttt{\makefirstuc} if the field requires a case-change.
The reason the \texttt{description} field is chosen for the modification is because the first field listed in \texttt{⟨list1⟩} of \texttt{dual-entry-map} is the \texttt{name} field which maps to \texttt{description} (the first field in the second list \texttt{⟨list2⟩}). This means that the hyperlink for the dual entry should be put in the \texttt{description} field.

For the primary entry, the \texttt{name} field is looked up in the second list from the \texttt{dual-entry-map} setting. This is the third item in this second list, so the third item in the first list is selected, which also happens to be the \texttt{description} field, so the hyperlink for the primary entry is put in the \texttt{description} field.

\texttt{dual-abbrv-backlink=\{(boolean)\}}

This is analogous to \texttt{dual-entry-backlink} but for entries defined with \texttt{@dualabbreviation} instead of \texttt{@dualentry}.

\texttt{dual-symbol-backlink=\{(boolean)\}}

This is analogous to \texttt{dual-entry-backlink} but for entries defined with \texttt{@dualsymbol} instead of \texttt{@dualentry}.

\texttt{dual-abbrventry-backlink=\{(boolean)\}}

Analogous to \texttt{dual-entry-backlink} but for entries defined with \texttt{@dualabbreviation-entry} instead of \texttt{@dualentry}. This setting can be problematic as the backlinks rely on the relevant field being known to \texttt{bib2gls}. Since the abbreviation style typically sets the \texttt{name} field (and sometimes the \texttt{description} field as well), you may find that no backlink appears. A simple workaround is to use \texttt{dual-field} (or \texttt{dual-field=dual}) to store the dual label in the \texttt{dual} field, and then use a style that checks for this field and adds the backlink.

With \texttt{glossaries-extra v1.30+} you can use:

\texttt{\GlsXtrDualBackLink\{(text)\}\{(label)\}}

which encapsulates \texttt{(text)} with a hyperlink to the dual. The \texttt{(label)} identifies the entry that requires a backlink. The dual’s label is obtained from the field given by:

\texttt{\GlsXtrDualField}

which defaults to \texttt{dual}. Note that if you assign a different field label with \texttt{dual-field}, then you will need to redefine \texttt{\GlsXtrDualField} as appropriate.

For example:

\texttt{\renewcommand*{\glsuserdescription}[2]{\%}
  \GlsXtrDualBackLink{\glslonguserfont{#1}}{#2}\%}
\setabbreviationstyle{long-short-user}
\GlsXtrLoadResources[src={entries},dual-field]
5.15 Tertiary Entries

dual-entryabbrv-backlink=\{\langle boolean\rangle\}

As dual-abbrventry-backlink but for entries defined with @dualentryabbreviation instead of @dualabbreviationentry.

dual-indexentry-backlink=\{\langle boolean\rangle\}

This is analogous to dual-entry-backlink but for entries defined with @dualindexentry instead of @dualentry.

dual-indexsymbol-backlink=\{\langle boolean\rangle\}

This is analogous to dual-entry-backlink but for entries defined with @dualindexsymbol and @dualindexnumber.

dual-indexabbrv-backlink=\{\langle boolean\rangle\}

This is analogous to dual-entry-backlink but for entries defined with @dualindexabbreviation and @tertiaryindexabbreviationentry.

dual-backlink=\{\langle boolean\rangle\}

Shortcut for:

dual-entry-backlink=\{\langle boolean\rangle\},
dual-abbrventry-backlink=\{\langle boolean\rangle\},
dual-abbrv-backlink=\{\langle boolean\rangle\},
dual-symbol-backlink=\{\langle boolean\rangle\},
dual-indexentry-backlink=\{\langle boolean\rangle\},
dual-indexsymbol-backlink=\{\langle boolean\rangle\},
dual-indexabbrv-backlink=\{\langle boolean\rangle\}

5.15 Tertiary Entries

tertiary-prefix=\{\langle value\rangle\}

This option indicates the prefix to use for the tertiary entries. The default value is tertiary. (including the terminating period).

As from version 1.8, the tertiary label prefix is identified in the .glistex file with:

\bibglsttertiaryprefixlabel\{\langle prefix\rangle\}

tertiary-type=\{\langle value\rangle\}

This option indicates that the tertiary entries should have their type field set to \langle value\rangle. If \langle value\rangle is empty the type is left unchanged. Unlike the type and dual-type options, there are no recognised keywords.
tertiary-category={⟨value⟩}

This option indicates that the tertiary entries should have their category field set to ⟨value⟩. If ⟨value⟩ is empty the category is left unchanged. Unlike the category and dual-category options, there are no recognised keywords.

5.16 Compound (Combined or Multi) Entries

These options refer to compound entries which are either defined in a .bib file with @compoundset or are defined in the document using \multiglossaryentry (or \provideglossaryentry). See section 4.10 for further details.

compound-options-global={⟨boolean⟩}

This is a boolean option. The default is compound-options-global={true}. If true, the compound entry options described in this section, except for compound-write-def, pick up all compound entries provided in the document (defined either with @compoundset or in the document with \multiglossaryentry).

If false, options only apply to compound entries defined with @compoundset in the current resource set. If cross-resource reference are disabled then any instances of @compoundset in subsequent resource sets can only be picked up from the .aux file on the next build.

compound-dependent={⟨boolean⟩}

This is a boolean option. The default is compound-dependent={false}. If you have chosen to switch off indexing for the other labels then they may not be selected (unless they have been indexed via another method, such as explicitly using \gls). You can use this option to make the other labels dependencies of the main label (if the entry given by main label is present in the current resource set).

This means that if the main label is selected then the other labels should also be selected (if dependencies are part of the selection criteria).

compound-add-hierarchy={⟨boolean⟩}

This is a boolean option. The default is compound-add-hierarchy={false}. If true, this will set the parent field for each element $e_i$ to the previous element $e_{i-1}$ in the list, provided that:

- the element $e_i$ isn’t the first element in the list;
- the element $e_i$ and the previous element $e_{i-1}$ are present in the current resource set;
- the element $e_i$ doesn’t already have the parent field set;
5.16 Compound (Combined or Multi) Entries

- the element \( e_i \) isn’t an ancestor of the previous element \( e_{i-1} \).
- the previous element \( e_{i-1} \) isn’t an ancestor of the element \( e_i \).

For example, if the .bib file contains:

```latex
@abbreviation{clostridium,
  short={C.},
  long={Clostridium}
}
@index{botulinum}
@compoundset{cbot,
  elements={clostridium,botulinum}}
```

Then the botulinum entry will have its parent field set to clostridium. The clostridium entry won’t be adjusted (since it’s the first element in the list).

**compound-has-records={⟨boolean⟩}**

This option may take one of the following values:

**true** Any compound entry referenced with commands like \texttt{\textbackslash mgls} is considered to have records for each element for selection purposes, even if there are no records in the .aux file. (This is useful on the first \LaTeX{} run where the compound entries are defined with \texttt{@compoundset}.)

**false** The element records are created as they normally are with commands like \texttt{\textbackslash gls} that are internally used by \texttt{\textbackslash mgls}.

**default** Behaves like \texttt{compound-has-records={true}} if the current resource set has any .bib files containing one or more \texttt{@compoundset} entry types. Otherwise behaves like \texttt{compound-has-records={false}}.

The default is \texttt{compound-has-records={default}}. If the value is omitted, \texttt{true} is assumed.

**compound-adjust-name={⟨value⟩}**

If an entry has been identified as the main label in any compound entries then the name field can be adjusted with this option. Allowed values:

- \texttt{false} don’t adjust the name field (default);
- \texttt{unique} only adjust the name field if the entry is the main label of exactly one set;
- \texttt{once} adjust the name field if the entry is the main label of any set. Only one adjustment is made. If the entry is the main label of multiple compound entries there’s no guarantee which set will be chosen for the adjustment.
If the value isn’t supplied, once is assumed. As with name-case-change, the pre-adjusted name value will be copied to the text field provided the text field hasn’t already been set and provided that the entry isn’t an abbreviation.

The adjusted value will be in the form:

\glsxtrmultientryadjustedname{(sublist1)}{(name)}{(sublist2)}{mlabel}

where ⟨label⟩ is the compound entry label, ⟨name⟩ was the value of the name field before the adjustment, ⟨sublist1⟩ is the list of other labels before the main label (which will be empty if the main label is the first element in the set) and ⟨sublist2⟩ is the list of other labels after the main label (which will be empty if the main label is the last element in the set).

The adjustment is made before name-case-change (if set). The control sequence case changes (such as name-case-change={firstuc-cs}) will replace \glsxtrmultientryadjustedname with the relevant command (\Glsxtrmultientryadjustedname for firstuc-cs, \GlsXtrmultientryadjustedname for title-cs and \GLSxtrmultientryadjustedname for uc-cs).

\textbf{compound-main-type=⟨value⟩}

Set the type field of the main entries. The ⟨value⟩ is required and should be one of:

- same as entry: sets the type to the entry type (lower case and without the initial @). For example, if the entry was defined with @index, the type will be set to index. If you’ve used entry-type-aliases, this refers to the target entry type not the original entry type provided in the .bib file.

- same as original entry: set the type field to the original entry type (lower case and without the initial @) before it was aliased (behaves like same as entry if the entry type wasn’t aliased).

- same as base: sets the type to the base name of the .bib file (without the extension) that provided the entry definition;

- same as category: sets the type to the same as the category field;

- same as parent: sets the type to the same as the entry’s parent. If the entry doesn’t have a parent or if the parent doesn’t have the type field set, then no change is made.

- ⟨label⟩: sets the type field to ⟨label⟩.

This setting is governed by compound-type-override.

\textbf{compound-other-type=⟨value⟩}

Set the type field of the other entries. The ⟨value⟩ is required and should be one of:

- same as main: sets the type to the same as the main entry.
5.16 Compound (Combined or Multi) Entries

- **same as entry**: sets the `type` to the entry type (lower case and without the initial @). For example, if the entry was defined with `@index`, the `type` will be set to `index`. If you’ve used `entry-type-aliases`, this refers to the target entry type not the original entry type provided in the `.bib` file.

- **same as original entry**: sets the `type` field to the original entry type (lower case and without the initial @) before it was aliased (behaves like `same as entry` if the entry type wasn’t aliased).

- **same as base**: sets the `type` to the base name of the `.bib` file (without the extension) that provided the entry definition;

- **same as category**: sets the `type` to the same as the `category` field;

- **same as parent**: sets the `type` to the same as the entry’s parent. If the entry doesn’t have a parent or if the parent doesn’t have the `type` field set, then no change is made.

- **⟨label⟩**: sets the `type` field to `⟨label⟩`.

  This setting is governed by `compound-type-override`.

  `{\texttt{compound-type-override=}⟨/ boolean⟩}\}

  This is a boolean option. The default is `compound-type-override={false}`.

  If `true`, then the options `compound-main-type` and `compound-other-type` will overwrite the `type` field otherwise those options will only set the `type` field if it hasn’t already been set.

  `{\texttt{compound-write-def=}⟨value⟩}\}

  If compound entries are defined in the `.bib` files using `@compoundset`, this option governs whether or not to write their definition to the `.glstex` file. The value may be one of:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Don’t write the definitions to the <code>.glstex</code> file. For example, if you are</td>
</tr>
<tr>
<td></td>
<td>reloading a <code>.bib</code> file from another resource set, you will need this option</td>
</tr>
<tr>
<td></td>
<td>to prevent duplicate definitions. (The alternative is to define \bibglsdefcompoundset to use \providemultiglossaryentry instead of \multiglossaryentry.)</td>
</tr>
<tr>
<td>all</td>
<td>Write all definitions to the <code>.glstex</code> file, regardless of whether or not</td>
</tr>
<tr>
<td></td>
<td>they have been referenced using commands like \mgls.</td>
</tr>
<tr>
<td>ref</td>
<td>Only write the definitions for compound entries that have been referenced</td>
</tr>
<tr>
<td></td>
<td>using commands like \mgls. (Default.)</td>
</tr>
</tbody>
</table>

  The compound entries are defined in the `.glstex` file with `\bibglsdefcompoundset`.

318
6 Provided Commands

When \texttt{bib2gls} creates the \texttt{.glstex} file, it writes some definitions for custom commands in the form \texttt{\bibgls...} which may be changed as required. The command definitions all use \texttt{\providecommand} which means that you can define the command with \texttt{\newcommand} before the resource file is loaded.

Note that if you try to redefine any of these commands after the resource file has been loaded with \texttt{\renewcommand}, you will get an error on the first \LaTeX{} run when the \texttt{.glstex} file doesn’t exist. You may prefer to use \texttt{\glsrenewcommand} instead, which will generate a warning instead of an error.

Since many of the commands are actually used within the \texttt{.glstex} file, it’s best to use \texttt{\newcommand} before the first resource set and \texttt{\renewcommand} between resource sets if adjustments are necessary.

6.1 Entry Definitions

This section lists the commands (\texttt{\bibglsnew...}) used to define entries. Note that the entry definition commands are actually used when \LaTeX{} inputs the resource file, so redefining them after the resource file is loaded won’t have an effect on the entries defined in that resource file (but will affect entries defined in subsequent resource files). Each provided command is defined in the \texttt{.glstex} file immediately before the first entry that requires it, so only the commands that are actually needed are provided.

The \texttt{sort} key may be set within the \texttt{.glstex} entry definition, but its value is usually not required in the document unless you are using a hybrid method with \texttt{record={hybrid}} (in which case, it’s redundant to get \texttt{bib2gls} to sort).

After each entry is defined, if it has any associated locations and the default \texttt{save-loclist} =\{\texttt{true}\} is set, then the locations are added using:

\begin{verbatim}
\glsxtrfieldlistadd{⟨label⟩}{⟨field⟩}{⟨item⟩}
\end{verbatim}

Any additional fields that don’t have associated keys are then set (if required) with \texttt{\GlsXtrSetField}.

\texttt{\bibglsnewentry}

\begin{verbatim}
\bibglsnewentry{⟨label⟩}{⟨options⟩}{⟨name⟩}{⟨description⟩}
\end{verbatim}

This command is used to define terms identified with the \texttt{@entry} type. The definition provided in the \texttt{.glstex} file is:
6.1 Entry Definitions

This uses the starred form \longnewglossaryentry* that doesn’t automatically append \nopostdesc (which interferes with the post-description hooks provided by category attributes).

\bibglsnewsymbol

This command is used to define terms identified with the @symbol type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewsymbol}[4]{%\longnewglossaryentry*{#1}{name={#3},sort={#1},category={symbol},#2}{#4}%}

Note that this sets the sort field to the label, but this may be overridden by the \langle options \rangle if the sort field was supplied or if bib2gls has determined the value whilst sorting the entries.

This also sets the category to symbol, but again this may be overridden by \langle options \rangle if the entry had the category field set in the .bib file or if the category was overridden with category=\langle value \rangle.

\bibglsnewnumber

This command is used to define terms identified with the @number type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewnumber}[4]{%\longnewglossaryentry*{#1}{name={#3},sort={#1},category={number},#2}{#4}%}

This is much the same as \bibglsnewsymbol above but sets the category to number. Again the sort and category keys may be overridden by \langle options \rangle.
6.1 Entry Definitions

\bibglsnewindex

This command is used to define terms identified with the \texttt{@index} type. The definition provided in the \texttt{.glstex} file is:

\begin{verbatim}
\providecommand*\bibglsnewindex[2]{\
\newglossaryentry{#1}{name={#1},category={index},description={},#2}}
\end{verbatim}

This makes the \texttt{name} default to the \texttt{(label)}, assigns the \texttt{category} to index and sets an empty \texttt{description}. These settings may be overridden by \texttt{(options)}.

Note that the \texttt{description} doesn’t include \texttt{\nopostdesc} to allow for the post-description hook used by category attributes.

\bibglsnewindexplural

This command is used to define terms identified with the \texttt{@indexplural} type. The definition provided in the \texttt{.glstex} file is:

\begin{verbatim}
\providecommand\bibglsnewindexplural[3]{\
\newglossaryentry{#1}{name={#3},category={indexplural},description={},#2}}
\end{verbatim}

This assigns the \texttt{category} to indexplural and sets an empty \texttt{description}. These settings may be overridden by \texttt{(options)}.

\bibglsnewabbreviation

This command is used to define terms identified with the \texttt{@abbreviation} type. The definition provided in the \texttt{.glstex} file is:

\begin{verbatim}
\providecommand\bibglsnewabbreviation[4]{\
\newabbreviation[#2]{#1}{#3}{#4}}
\end{verbatim}

Since this uses \texttt{\newabbreviation}, it obeys the abbreviation style for its given \texttt{category} (which may have been set in \texttt{(options)}, either from the \texttt{category} field in the \texttt{.bib} file or through the \texttt{category} option). Similarly the \texttt{type} will obey \texttt{\glsxtrabbrvtype} unless the value is supplied in the \texttt{.bib} file or through the \texttt{type} option.
6.1 Entry Definitions

\bibglsnewacronym

This command is used to define terms identified with the \acronym type. The definition provided in the .glstex file is:
\providecommand{\bibglsnewacronym}[4]{\newacronym[#2]{#1}{#3}{#4}}

This works in much the same way as \bibglsnewabbreviation. Remember that with the glossaries-extra package \newacronym is redefined to just use \newabbreviation with the default type set to \acrshort and the default category set to acronym.

\bibglsnewdualentry

This command is used to define terms identified with the \dualentry type. The definition provided in the .glstex file is:
\providecommand{\bibglsnewdualentry}[4]{\longnewglossaryentry*{#1}{name={#3},#2}{#4}}

\bibglsnewdualindexentry

This command is used to define primary terms identified with the \dualindexentry type. The definition provided in the .glstex file is:
\providecommand{\bibglsnewdualindexentry}[4]{\longnewglossaryentry*{#1}{name={#3},category={index},#2}{}}

Note that this definition ignores the \description argument.

\bibglsnewdualindexentrysecondary

This command is used to define secondary terms identified with the \dualindexentry type. The definition provided in the .glstex file is:
\providecommand{\bibglsnewdualindexentrysecondary}[4]{\longnewglossaryentry*{#1}{name={#3},#2}{#4}}
6.1 Entry Definitions

\bibglsnewdualindexsymbol

This command is used to define primary terms identified with the \dualindexsymbol type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewdualindexsymbol}[5]{%
  \longnewglossaryentry*{#1}{name={#3},category={index},symbol={#4},#2}
}%

Note that this definition ignores the \textit{description} argument.

\bibglsnewdualindexsymbolsecondary

This command is used to define secondary terms identified with the \dualindexsymbol type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewdualindexsymbolsecondary}[5]{%
  \longnewglossaryentry*{#1}{name={#3},category={symbol},symbol={#4},#2}
}%

\bibglsnewdualindexnumber

This command is used to define primary terms identified with the \dualindexnumber type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewdualindexnumber}[5]{%
  \longnewglossaryentry*{#1}{name={#3},category={index},symbol={#4},#2}
}%

Note that this definition ignores the \textit{description} argument.

\bibglsnewdualindexnumbersecondary

This command is used to define secondary terms identified with the \dualindexnumber type. The definition provided in the .glstex file is:
6.1 Entry Definitions

\providecommand{\bibglsnewdualindexnumbersecondary}{% 
\longnewglossaryentry@{#1}{name={#3},category={number},symbol={#4},#2} 
{#5}%
}

\bibglsnewdualindexabbreviation

\providecommand{\bibglsnewdualindexabbreviation}{⟨label⟩{⟨dual-label⟩}{⟨options⟩}{⟨name⟩} 
{⟨short⟩}{⟨long⟩}{⟨description⟩}
}

This command is used to define primary terms identified with the @dualindexabbreviation type. The default definition provided in the .glstex file is:

\providecommand{\bibglsnewdualindexabbreviation}{% 
\longnewglossaryentry@{#1}{% 
  name={\protect\bibglsuseabbrvfont{#4}{\glscategory{#2}}},% 
  category={index},#3{% 
}
}

In this case ⟨dual-label⟩ is the dual entry’s label, which is used to fetch the category label in \bibglsuseabbrvfont. (The category field for the dual isn’t used since a custom definition of \bibglsnewdualindexabbreviationsecondary may override the value known to bib2gls.)

Note that (as shown above) with the default abbreviation-name-fallback={short} the name uses:

\bibglsuseabbrvfont{⟨text⟩}{⟨category⟩}

to format the name, which ensures that it uses the same font as the short form for the dual abbreviation. This will use \bibglsuseabbrvfont if it’s defined otherwise it will be defined to replicate that command. If abbreviation-name-fallback is set to some other field then the name uses:

\bibglsuselongfont{⟨text⟩}{⟨category⟩}

instead, which ensures that it uses the same font as the long form for the dual abbreviation.

\bibglsnewdualindexabbreviationsecondary

\providecommand{\bibglsnewdualindexabbreviationsecondary}{⟨label⟩{⟨options⟩}{⟨name⟩} 
{⟨short⟩}{⟨long⟩}{⟨description⟩}
}

This command is used to define secondary terms identified with the @dualindexabbreviation type. The definition provided in the .glstex file is:
6.1 Entry Definitions

\providecommand{\bibgsnewdualindexabbreviationsecondary}[6]{% 
  \ifstrempty{#6}%
  {\newabbreviation[#2]{#1}{#4}{#5}%
   {\newabbreviation[#2,description={#6}]{#1}{#4}{#5}}%
  }
}\bibgsnewdualabbreviationentry

\bibgsnewdualabbreviationentry{⟨label⟩}{⟨options⟩}{⟨short⟩}{⟨long⟩}
{⟨description⟩}

This command is used to define primary terms identified with the \texttt{@dualabbreviation-entry} type. The definition provided in the .glstex file is:

\providecommand{\bibgsnewdualabbreviationentry}{[5]{% 
  \newabbreviation[#2]{#1}{#3}{#4}%
  }
}\bibgsnewdualabbreviationentrysecondary

\bibgsnewdualabbreviationentrysecondary{⟨label⟩}{⟨options⟩}{⟨short⟩}
{⟨long⟩}{⟨description⟩}

This command is used to define secondary terms identified with the \texttt{@dualabbreviation-entry} type. The definition provided in the .glstex file is:

\providecommand{\bibgsnewdualabbreviationentrysecondary}{[5]{% 
  \longnewglossaryentry*{#1}{#2}{#5}%
  }
}\bibgsnewdualabbreviationentryabbreviation

\bibgsnewdualabbreviationentryabbreviation{⟨label⟩}{⟨options⟩}{⟨short⟩}{⟨long⟩}
{⟨description⟩}

This command is used to define primary terms identified with the (now deprecated) entry type \texttt{@dualentryabbreviation}. The definition provided in the .glstex file is:
Note that this definition ignores the \textit{description} argument.

\bibglsnewdualentryabbreviationsecondary

This command is used to define secondary terms identified with the (now deprecated) entry type \texttt{@dualentryabbreviation}. The definition provided in the .gls\TeX\ file is:

\providecommand{\bibglsnewdualentryabbreviationsecondary}[5]{% 
 \longnewglossaryentry*{#1}{#2}{#5}\}

Note that this definition ignores the \textit{short} and \textit{long} arguments (which will typically be empty unless the default mappings are changed).

\bibglsnewdualsymbol

This command is used to define terms identified with the \texttt{@dualsymbol} type. The definition provided in the .gls\TeX\ file is:

\providecommand{\bibglsnewdualsymbol}[4]{% 
 \longnewglossaryentry*{#1}{name={#3},sort={#1},category={symbol},#2}\}

\bibglsnewdualnumber

This command is used to define terms identified with the \texttt{@dualnumber} type. The definition provided in the .gls\TeX\ file is:

\providecommand{\bibglsnewdualnumber}[4]{% 
 \longnewglossaryentry*{#1}{name={#3},sort={#1},category={symbol},#2}\}
6.1 Entry Definitions

\bibglsnewdualabbreviation

\bibglsnewdualabbreviation{⟨label⟩}{⟨options⟩}{⟨short⟩}{⟨long⟩}

This command is used to define terms identified with the @dualabbreviation type where the duallong field is swapped with the long field and the dualshort field is swapped with the short field. The definition provided in the .glstex file is:

\providecommand{\bibglsnewdualabbreviation}[4]{
  \newabbreviation[#2]{#1}{#3}{#4}%
}

\bibglsnewdualacronym

\bibglsnewdualacronym{⟨label⟩}{⟨options⟩}{⟨short⟩}{⟨long⟩}

This command is used to define terms identified with the @dualacronym type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewdualacronym}[4]{
  \newacronym[#2]{#1}{#3}{#4}%
}

This works in much the same way as \bibglsnewdualabbreviation. Remember that with the glossaries-extra package \newacronym is redefined to just use \newabbreviation with the default type set to acroymtype and the default category set to acronym.

\bibglsnewtertiaryindexabbreviationentry

\bibglsnewtertiaryindexabbreviationentry{⟨label⟩}{⟨dual-label⟩}{⟨options⟩}{⟨name⟩}{⟨short⟩}{⟨long⟩}{⟨description⟩}

This is used to define primary terms identified with the @tertiaryindexabbreviationentry type. It’s essentially the same as \bibglsnewdualindexabbreviation. The definition provided in the .glstex file is:

\providecommand{\bibglsnewtertiaryindexabbreviationentry}[7]{
  \longnewglossaryentry*{#1}{%
    name={\protect\bibglsuseabbrvfont{#4}{\glscategory{#2}}},%
    category={index},#3%
  }
}
6.1 Entry Definitions

\bibglsnewtertiaryindexabbreviationentrysecondary

\bibglsnewtertiaryindexabbreviationentrysecondary\{\langle label\rangle\}\{\langle tertiary-label\rangle\}\{\langle options\rangle\}\{\langle tertiary-opts\rangle\}\{\langle primary-name\rangle\}\{\langle short\rangle\}\{\langle long\rangle\}\{\langle description\rangle\}

This command is used to define both the secondary and tertiary terms identified with the \@tertiaryindexabbreviationentry type. The secondary term is an abbreviation and the tertiary term is a regular entry. The definition written to the .glstex file is:

\providecommand{\bibglsnewtertiaryindexabbreviationentrysecondary}\[8\]{\%
  \newabbreviation[#3]{#1}{#6}{#7}\%
  \longnewglossaryentry*{#2}\%
  \name={\protect\bibglsuselongfont{#7}{\glscategory{#1}}},#4\%
  \#8\%}
}

The \langle label\rangle is the label for the secondary (abbreviation) entry and \langle tertiary-label\rangle is the label for the tertiary (regular) entry. The fifth argument (\langle primary name\rangle) isn’t used but is provided if required for a custom redefinition. The name field for the tertiary is obtained from the \langle long\rangle argument encapsulated by \bibglsuselongfont to format the name, which ensures that it uses the same font as the long form for the dual abbreviation. This will use \globuselongfont if it’s defined otherwise it will be defined to replicate that command.

\bibglsnewbibtexentry

\bibglsnewbibtexentry\{\langle label\rangle\}\{\langle options\rangle\}\{\langle name\rangle\}\{\langle description\rangle\}

This command is used to define the main term identified with \@bibtexentry. The definition written to the .glstex file is:

\providecommand{\bibglsnewbibtexentry}\[4\]{\%
  \longnewglossaryentry*{#1}{name={#3},#2}\%}
}

\bibglsnewcontributor

\bibglsnewcontributor\{\langle label\rangle\}\{\langle options\rangle\}\{\langle name\rangle\}\{\langle description\rangle\}

This command is used to define terms identified with \@contributor (typically implicitly created through \@bibtexentry). The definition written to the .glstex file is:

\providecommand{\bibglsnewcontributor}\[4\]{\%
  \longnewglossaryentry*{#1}{name={#3},#2}\%}
}
6.1 Entry Definitions

\bibglsnewprogenitor

\bibglsnewprogenitor\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}

This command is used to define the main terms created by \texttt{@progenitor}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewprogenitor}\[4\]{
\longnewglossaryentry*{#1}{name={#3},#2}{#4}\
}

\bibglsnewspawnindex

\bibglsnewspawnindex\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}

This command is used to define the main terms created by \texttt{@spawnindex}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewspawnindex}\[4\]{
\longnewglossaryentry*{#1}{name={#3},#2}{#4}\
}

\bibglsnewspawnedindex

\bibglsnewspawnedindex\{⟨label⟩\}{⟨options⟩}

This command is used to define the terms spawned by \texttt{@progenitor} or \texttt{@spawnindex}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewspawnedindex}\[2\]{
\newglossaryentry{#1}{name={#1},category=index,description={},#2}\
}

\bibglsnewspawnindexplural

\bibglsnewspawnindexplural\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}

This command is used to define the main terms created by \texttt{@spawnindexplural}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewspawnindexplural}\[4\]{
\longnewglossaryentry*{#1}{name={#3},#2}{#4}\
}
6.1 Entry Definitions

\bibglsnewspawnedindexplural

This command is used to define the terms spawned by \texttt{@spawnindexplural}. The definition is written to the \texttt{.glstex} file as:

\providecommand{\bibglsnewspawnedindexplural}[3]{\newglossaryentry{#1}{name={#3},category={indexplural},description=\},#2}%%

\bibglsnewspawnentry

This command is used to define the main terms created by \texttt{@spawnentry}. The definition is written to the \texttt{.glstex} file as:

\providecommand{\bibglsnewspawnentry}[4]{\newglossaryentry*{#1}{name={#3},#2}{#4}%%

\bibglsnewspawnedentry

This command is used to define the terms spawned by \texttt{@spawnentry}. The definition is written to the \texttt{.glstex} file as:

\providecommand{\bibglsnewspawnedentry}[4]{\newglossaryentry*{#1}{name={#3},#2}{#4}%%

\bibglsnewspawnabbreviation

This command is used to define the main terms created by \texttt{@spawnabbreviation}. The definition is written to the \texttt{.glstex} file as:

\providecommand{\bibglsnewspawnabbreviation}[4]{\newabbreviation[#2]{#1}{#3}{#4}%%
6.1 Entry Definitions

\bibglsnewspawnedabbreviation

\bibglsnewspawnedabbreviation{(label)}{(options)}{(short)}{(long)}

This command is used to define the terms spawned by \@spawnabbreviation. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawnedabbreviation}[4]{{}
  \newabbreviation[#2]{#1}{#3}{#4}{}
}

\bibglsnewspawnacronym

\bibglsnewspawnacronym{(label)}{(options)}{(short)}{(long)}

This command is used to define the main terms created by \@spawnacronym. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawnacronym}[4]{{}
  \newacronym[#2]{#1}{#3}{#4}{}
}

\bibglsnewspawnedacronym

\bibglsnewspawnedacronym{(label)}{(options)}{(short)}{(long)}

This command is used to define the terms spawned by \@spawnacronym. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawnedacronym}[4]{{}
  \newacronym[#2]{#1}{#3}{#4}{}
}

\bibglsnewspawnsymbol

\bibglsnewspawnsymbol{(label)}{(options)}{(name)}{(description)}

This command is used to define the main terms created by \@spawnsymbol. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawnsymbol}[4]{{}
  \longnewglossaryentry*{#1}{name={#3},#2}{#4}{}
}
6.1 Entry Definitions

\bibglsnewspawnedsymbol

\bibglsnewspawnedsymbol\{\langle\text{label}\rangle\}\{\langle\text{options}\rangle\}\{\langle\text{name}\rangle\}\{\langle\text{description}\rangle\}

This command is used to define the terms spawned by @spawnsymbol. The definition is written to the .glistex file as:

\providecommand{\bibglsnewspawnedsymbol}[4]{% \longnewglossaryentry*{#1}{name={#3},sort={#1},category={spawnedsymbol},#2}{#4}}

\bibglsnewspawnnumber

\bibglsnewspawnnumber\{\langle\text{label}\rangle\}\{\langle\text{options}\rangle\}\{\langle\text{name}\rangle\}\{\langle\text{description}\rangle\}

This command is used to define the main terms created by @spawnnumber. The definition is written to the .glistex file as:

\providecommand{\bibglsnewspawnnumber}[4]{% \longnewglossaryentry*{#1}{name={#3},#2}{#4}}

\bibglsnewspawnednumber

\bibglsnewspawnednumber\{\langle\text{label}\rangle\}\{\langle\text{options}\rangle\}\{\langle\text{name}\rangle\}\{\langle\text{description}\rangle\}

This command is used to define the terms spawned by @spawnnumber. The definition is written to the .glistex file as:

\providecommand{\bibglsnewspawnednumber}[4]{% \longnewglossaryentry*{#1}{name={#3},sort={#1},category={spawnednumber},#2}{#4}}

\bibglsnewspawndualindexentry

\bibglsnewspawndualindexentry\{\langle\text{label}\rangle\}\{\langle\text{options}\rangle\}\{\langle\text{name}\rangle\}\{\langle\text{description}\rangle\}

This command is used to define the progenitor’s primary term created by @spawndualindexentry. The definition is written to the .glistex file as:

\providecommand{\bibglsnewspawndualindexentry}[4]{% \longnewglossaryentry*{#1}{name={#3},category={index},#2}{#4}}

The \langle description\rangle argument is ignored.
\bibglsnewspawndualindexentrysecondary

\bibglsnewspawndualindexentrysecondary\{\langle label\rangle\}\{\langle options\rangle\}\{\langle name\rangle\}\{\langle description\rangle\}

This command is used to define the progenitor’s secondary (dual) term created by \@spawn-dualindexentry. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawndualindexentrysecondary}[4]{% 
\longnewglossaryentry*{#1}{name={#3},#2}{#4}%
}

6.2 Compound Entry Sets

\bibglsdefcompoundset

\bibglsdefcompoundset\{\langle options\rangle\}\{\langle label\rangle\}\{\langle main\rangle\}\{\langle elements\rangle\}

The default definition uses \multiglossaryentry.

6.3 Location Lists and Cross-References

These commands deal with the way the location lists and cross references are formatted. The commands typically aren’t used until the entry information is displayed in the glossary, so you may redefine these commands after the resource file has been loaded.

\bibglsseesep

\bibglsseesep

Any entries that provide a see field (and that field hasn’t be omitted from the location list with see={omit}) will have \bibglsseesep inserted between the see part and the location list (unless there are no locations, in which case just the see part is displayed without \bibglsseesep).

This command is provided with:

\providecommand{\bibglsseesep}{, } 
You can define this before you load the .bib file:

\newcommand{\bibglsseesep}{; } 
\GlsXtrLoadResources[src={entries}]

Or you can redefine it afterwards:

\GlsXtrLoadResources[src={entries}]
\glsrenewcommand{\bibglsseesep}{; }
\bibglsseealso\sep
\bibglsseealso\sep
This is like \bibglssee\sep but is used with cross-reference lists provided with the seealso field, if supported.

\bibglsalias\sep
\bibglsalias\sep
This is like \bibglssee\sep but is used with cross-reference lists provided with the alias field.

\bibglsuse\see\sep
\bibglsuse\see\sep\{\langle\textit{label}\rangle}\}
Displays the formatted cross-reference list stored in the see field for the given entry. This just defaults to \glsxtruse\see\{\langle\textit{label}\rangle}\}.

\bibglsuse\seealso\sep
\bibglsuse\seealso\sep\{\langle\textit{label}\rangle}\}
Displays the formatted cross-reference list stored in the seealso field for the given entry. This just defaults to \glsxtruse\seealso\{\langle\textit{label}\rangle}\}.

\bibglsuse\alias\sep
\bibglsuse\alias\sep\{\langle\textit{label}\rangle}\}
Displays the formatted cross-reference stored in the alias field for the given entry. This is defined to use \glsseeformat.

\bibglsdelimN\sep
\bibglsdelimN\sep
Separator between individual locations, except for the last. This defaults to \delimN.
6.3 Location Lists and Cross-References

\bibglslastDelimN

Separator between penultimate and final individual locations. This defaults to ,~ to discourage lonely locations.

\bibglsscompact

This command is used with compact-ranges when the end location in a range is compacted. The first argument ⟨pattern⟩ indicates the location pattern: digit for digits, roman for lower case Roman numerals, ROMAN for upper case Roman numerals and alpha for alphabetical locations. The actual location is split into two parts, ⟨part1⟩ and ⟨part2⟩. The string concatenation ⟨part1⟩⟨part2⟩ forms the actual location.

This just does ⟨part2⟩ by default.

\bibglspassim

If max-loc-diff is greater than 1, then any implicit ranges that have skipped over gaps will be followed by \bibglspassim, which is defined as:

\providecommand{\bibglspassim}{\bibglspassimname}

You can define this before you load the .bib file:

\newcommand{\bibglspassim}{}
\GlsXtrLoadResources[src={entries}]

Or you can redefine it afterwards:

\GlsXtrLoadResources[src={entries}]
\glsrenewcommand{\bibglspassim}{}

\bibglspassimname

The default definition is obtained from the language resource file. For example, with bib2gls-en.xml the provided definition is:

\providecommand{\bibglspassimname}{passim}
Explicit ranges formed using \texttt{format=}{{} and \texttt{format=}{{}} or \texttt{format=}{{(}}\texttt{csname})\texttt{)} (where \texttt{csname} matches and is a text-block command without the initial backslash) in the optional argument of commands like \texttt{gls} or \texttt{glsadd} are encapsulated within the argument of \texttt{bibglsrange}. By default, this simply does its argument. This command is not used with implicit ranges that are formed by collating consecutive locations or when \texttt{merge-ranges=}{{true}} is used.

If an explicit ranges conflicts with a record, a warning will be issued and the conflicting record (the interloper) will be shifted to the front of the range inside the argument of \texttt{bibglsinterloper}. The default definition just does \texttt{⟨location\bibglsdelimN} so that it fits neatly into the list.

For example, suppose on page 4 of my document I start a range with:

\texttt{\glsadd[format=}{{}}}\texttt{sample}

and end it on page 9 with:

\texttt{\glsadd[format=}{{}}}\texttt{sample}

This forms an explicit range, but let’s suppose on page 6 I have:

\texttt{\gls[format=}{{hyperbf}}}\texttt{sample}

This record conflicts with the explicit range (which doesn’t include \texttt{hyperbf} in the format). This causes a warning and the conflicting entry will be moved before the start of the explicit range resulting in \texttt{6, 4–9}.

Note that implicit ranges can’t be formed from interlopers (nor can implicit ranges be merged with explicit ones with the default \texttt{merge-ranges=}{{false}}), so if \texttt{\gls[format=}{{hyperbf}}}\texttt{sample} also occurs on pages 7 and 8 then the result will be \texttt{6, 7, 8, 4–9}. Either remove the explicit range or remove the conflicting entries. (Alternatively, redefine \texttt{\bibglsinterloper} to ignore its argument, which will discard the conflicting entries.)

If the \texttt{loc-prefix} option is on, \texttt{\bibglspostlocprefix} will be inserted at the start of location lists, and its default definition includes \texttt{\bibglspostlocprefix} placed after the prefix text. This command is provided with:


\providecommand{\bibglspostlocprefix}{\␣}

which puts a space between the prefix text and the location list. You can define this before you load the .bib file:

\newcommand{\bibglspostlocprefix}{: }
\GlsXtrLoadResources[src={entries},loc-prefix]

Or you can redefine it afterwards:

\GlsXtrLoadResources[src={entries},loc-prefix]
\glsrenewcommand{\bibglspostlocprefix}{: }

\bibglslocprefix

\bibglslocprefix{(n)}

If the loc-prefix option is on, this command will be provided. The location of the definition is determined by the loc-prefix-def option. For example, if the document has:

\GlsXtrLoadResources[type={main},loc-prefix-def={individual},loc-prefix=\{p.,pp.\},src={entries}]

then the following will be added to the .glstex file:

\apptoglossarypreamble[main]{%
\providecommand{\bibglslocprefix}[1]{%
\ifcase##1
\or p.\bibglslocprefix
\else pp.\bibglslocprefix
\fi
}%
}%

However, if the type key is missing or if the option loc-prefix-def={local} is used, then the following will be added instead:

\appto\glossarypreamble{%
\providecommand{\bibglslocprefix}[1]{%
\ifcase#1
\or p.\bibglslocprefix
\else pp.\bibglslocprefix
\fi
}%
}%

If loc-prefix-def={global} is used then the definition is global (outside of the glossary preamble).
6.3 Location Lists and Cross-References

\bibglspagename

If `loc-prefix={true}` is used, then this command is provided using the value of `tag.page` from the language resource file. For example with `bib2gls-en.xml` the definition is:
\providecommand{\bibglspagename}{Page}

\bibglspagename

If `loc-prefix={true}` is used, then this command is provided using the value of `tag.pages` from the language resource file. For example with `bib2gls-en.xml` the definition is:
\providecommand{\bibglspagesname}{Pages}

\bibglslocsuffix

If the `loc-suffix` option is on, this command will be provided. The location of the definition depends on the `loc-suffix-def` option.

This command’s definition depends on the value provided by `loc-suffix`. For example, with `loc-suffix={\@.}` the command is defined as:
\providecommand{\bibglslocsuffix}[1]{\@.}

(which ignores the argument).

Whereas with `loc-suffix={⟨A⟩,⟨B⟩,⟨C⟩}` the command is defined as:
\providecommand{\bibglslocsuffix}[1]{\ifcase#1 ⟨A⟩\or ⟨B⟩\else ⟨C⟩\fi}

Note that this is slightly different from `\bibglslocprefix` as it includes the 0 case, which in this instance means that there were no locations but there was a cross-reference. This command isn’t added when the location list is empty.

\bibglslocationgroup

When the `loc-counters` option is used, the locations for each entry are grouped together according to the counter (in the order specified in the value of `loc-counters`). Each group of locations is encapsulated within `\bibglslocationgroup`, where `⟨n⟩` is the number of locations within the group, `⟨counter⟩` is the counter name and `⟨list⟩` is the formatted location.
sub-list. By default, this simply does \langle list \rangle, but may be defined (before the resources are loaded) or redefined (after the resources are loaded) as required.

For example:

\newcommand*{\bibglslocationgroup}[3]{% 
  \ifnum#1=1
  #2:
  \else
  #2s:
  \fi
  #3%
}

\GlsXtrLoadResources[
  loc-counters={equation,page},% group locations by counter
  src={entries}% data in entries.bib
]

This will prefix each group with the counter name, if there's only one location, or the counter name followed by “s”, if there are multiple locations within the group.

There are various ways to adapt this to translate the counter name to a different textual label, such as:

\providecommand{\pagename}{Page}
\providecommand{\pagesname}{Pages}
\providecommand{\equationname}{Equation}
\providecommand{\equationsname}{Equations}

\newcommand*{\bibglslocationgroup}[3]{% 
  \ifnum#1=1
    \ifcsdef{#2name}{\csuse{#2name}}{#2}:
  \else
    \ifcsdef{#2sname}{\csuse{#2sname}}{#2s}:
    \fi
    \fi
  #3%
}

\bibglslocationgroupsep

When the \texttt{loc-counters} option is set, this command is used to separate each location sub-group. It may be defined before the resources are loaded:

\newcommand*{\bibglslocationgroupsep}{; }
6.3 Location Lists and Cross-References

\GlsXtrLoadResources[
  loc-counters={equation,page},% group locations by counter
  src={entries}% data in entries.bib
]

or redefined after the resources are loaded:

\GlsXtrLoadResources[
  loc-counters={equation,page},% group locations by counter
  src={entries}% data in entries.bib
]

\glsrenewcommand*{\bibglslocationgroupsep}{; }

\bibglsprimary

\bibglsprimary{(n)}{(locations)}

When the save-principal-locations option is used, the principal locations are stored in the primarylocations field encapsulated with this command, unless the locations are split into groups according to the location counter.

The first argument is the number of locations in the list. The second argument is the list of locations formatted in the usual way. The default definition is to ignore the first argument and simply do the second.

If the locations are split into groups, then \bibglsprimarylocationgroup is used for each group instead (separated with \bibglsprimarylocationgroupsep).

\bibglsprimarylocationgroup

\bibglsprimarylocationgroup{(n)}{(counter)}{(list)}

As \bibglslocationgroup but for primary group locations.

\bibglsprimarylocationgroupsep

\bibglsprimarylocationgroupsep

As \bibglslocationgroupsep but for primary group locations.
\bibglssupplemental

\bibglssupplemental{(n)}{(list)}

When the supplemental-locations option is used, the locations from a supplementary document are encapsulated within the \(\text{(list)}\) part of \texttt{\bibglssupplemental}. The first argument \(\langle n\rangle\) (ignored by default) is the number of supplementary locations.

If multiple supplemental sources are permitted (that is, \texttt{bib2gls} has detected that the document is using at least version 1.36 of \texttt{glossaries-extra}), then the \(\langle\text{list}\rangle\) part will consist of sub-lists for each external source. In this case, \(\langle n\rangle\) will be the total number of elements across all the sub-lists.

\bibglssupplementalsublist

\bibglssupplementalsublist{(n)}{(external document)}{(list)}

If multiple supplemental sources are permitted, this will be used to format each sub-list, where \(\langle n\rangle\) (ignored by default) is the number of elements in the sub-list, \(\langle\text{external document}\rangle\) (ignored by default) is the external source and \(\langle\text{list}\rangle\) is the list of supplementary locations in \(\langle\text{external document}\rangle\).

\bibglssupplementalsep

\bibglssupplementalsep

The separator between the main location list and the supplementary location list. By default this is just \texttt{\bibglsdelimN}. This may be defined before the resources are loaded:

\newcommand{\bibglssupplementalsep}{; }

\GlsXtrLoadResources[
  supplemental-locations={supplDoc},
  src={entries}]

or redefined after the resources are loaded:

\GlsXtrLoadResources[
  supplemental-locations={supplDoc},
  src={entries}]

\glsrenewcommand{\bibglssupplementalsep}{; }

341
6.4 Letter Groups

\bibglssupplementalsubsep

The separator between the supplementary location sub-lists. By default this is just \bibglsdelimN.

\bibglshrefchar

\bibglshrefchar{⟨hex⟩}{⟨char⟩}
Expands to a literal percent character followed by ⟨hex⟩. The second argument is ignored.

\bibglshrefunicode

\bibglshrefunicode{⟨hex⟩}{⟨char⟩}
Expands to the second argument. The first argument is ignored.

\bibglshexunicodechar

\bibglshexunicodechar{⟨hex⟩}
This command is used by the hex-unicode-fields option when replacing any Unicode characters. The argument ⟨hex⟩ is the hexadecimal character code. Note that the argument isn’t preceded by the double-quote character " (which is normally used to identify hexadecimal numbers in \LaTeX). Instead, the definition needs to insert this character, if appropriate.

If \bib2gls has detected that the hyperref package has been loaded, it will provide a definition that may be used in PDF bookmarks provided that hyperref’s unicode option is set. Otherwise the command will simply do \symbol{"⟨hex⟩} (which will require an appropriate font in order to render the symbol correctly).

6.4 Letter Groups

The commands listed in this section are provided for use with the --group switch and glossary styles that display the letter group title. If these need their definitions altered, they should be defined before the resource file is loaded if field expansion is on (--expand-fields) otherwise they may be redefined afterwards.

The base glossaries package determines group titles through a fairly simplistic rule. Both makeindex and xindy write the line:

\glsgroupheading{⟨label⟩}
6.4 Letter Groups

to the associated glossary file at the start of each new letter group. For example, the “A” letter group will be written as:

\glsgroupheading{A}

This is quite straightforward and the heading title can just be “A”. The “Symbols” group is written as:

\glsgroupheading{glssymbols}

To allow for easy translation, the base glossaries package has the simple rule:

- if \langle heading\rangle groupname exists use that;
- otherwise just use \langle heading\rangle.

There’s no \Agroupname provided, but \glssymbolsgroupname is provided and is supported by the associated language modules, such as glossaries-french. (Similarly for the “Numbers” group.)

The glossary styles that provide hyperlinks to the groups (such as indexhypergroup) use \langle heading\rangle to form the target name. A problem arises when active characters occur in \langle heading\rangle, which happens with extended characters and inputenc.

The glossaries-extra package (as from version 1.14) provides:

\glsxtrsetgrouptitle{\langle group label\rangle}{\langle group title\rangle}

to set the title for a group with the given label. The internal workings of \glsgroupheading are modified to use a slightly altered rule:

- if a title has been set using \glsxtrsetgrouptitle{\langle heading\rangle}{\langle title\rangle} for the given \langle heading\rangle, use that;
- if \langle heading\rangle groupname exists, use that;
- just use \langle heading\rangle for the title.

So if \glsxtrsetgrouptitle hasn’t been used, it falls back on the original rule.

The problem is now how to make the indexing application use the desired label in the argument of \glsgroupheading instead of selecting the heading based on the first character of each sort value for each top-level entry in that group. This can’t be done with makeindex, and with xindy it requires a custom language module, which isn’t a trivial task.

With bib2gls, a different approach is used. The .glstex file created isn’t comparable to the .gls file created by makeindex or xindy. There’s nowhere for bib2gls to write the \glsgroupheading line as it isn’t creating the code that typesets the glossary list. Instead it’s creating the code that defines the entries. The actual group heading is inserted by \printunsrtglossary and it’s only able to do this by checking if the entry has a group field and comparing it to the previous entry’s group field.
The behaviour of the group formation implemented by the sort methods may be changed with `group-formation`. With any setting other than `group-formation={default}`, the group label is set to `\bibglsunicodegroup{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}` and the title is set to `\bibglsunicodegrouptitle{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}` (see below) otherwise the label and title are determined by the sort method.

The collators used by the locale and letter-based rules save the following information for each entry based on the first significant letter of the `sort` field (if the letter is recognised as alphabetical, according to the rule):

- ⟨title⟩ The group’s title. This is typically title-cased. For example, if the rule recognises the digraph “dz”, then the title is “Dz”. Exceptions to this are included in the language resource file. If the key `grouptitle.case.⟨lc⟩` exists, where ⟨lc⟩ is the lower case version of ⟨title⟩, then the value of that key is used instead. For example, the Dutch digraph “ij” should be converted to “IJ”, so `bib2gls-en.xml` includes:

  <entry key="grouptitle.case.ij">IJ</entry>

  (See the --group switch for more details.)

- ⟨letter⟩ This is the actual letter at the start of the given entry’s `sort` field, which may be lower case or may contain diacritics that don’t appear in ⟨title⟩.

- ⟨id⟩ A numeric identifier. This may be the collation key or the code point for the given letter, depending on the sort method.

- ⟨type⟩ The entry’s glossary type. If not known, this will be empty. (bib2gls won’t know if you’ve modified the associated `\bibglsnew…` command to set the type. It can only know the type if it’s in the original .bib definition or is set using resource options such as type.)

The group field is then set using:

```
group=\bibglslettergroup{⟨title⟩}{⟨letter⟩}{⟨id⟩}{⟨type⟩}
```

This field needs to expand to a simple label, which `\bibglslettergroup` is designed to do. Note that non-letter groups are dealt with separately (see below).

\bibglssetlastgrouptitle

In the last resource (.glstex) file, after all the relevant group titles have been set with the commands listed below, there’s a final title setting:

```
\bibglssetlastgrouptitle{⟨cs⟩}{⟨specs⟩}
```

This does nothing by default, but the arguments are set to correspond to the group with the maximum id for that resource file. It’s provided as a convenient way of overriding the final group title without the inconvenience of looking up the group label in the .glstex
6.4 Letter Groups

file. If you have multiple glossaries or if you want to override a different group, then you need to inspect the .glstex file to work out the corresponding label (by finding the group assignment for one of the entries in that group).

The \langle cs \rangle argument is the control sequence used in the group field to obtain the label from \langle specs \rangle. For example, if the highest \langle id \rangle is 214748112 from:

\begin{verbatim}
group={\bibglsslettergroup\{Ø\}{Ø}\{214748112\}{}}
\end{verbatim}

then the last group is identified with:

\begin{verbatim}
\bibglssetlastgrouptitle{\bibglsslettergroup\{Ø\}{Ø}\{214748112\}{}}
\end{verbatim}

In this case \langle cs \rangle is \bibglsslettergroup and \langle specs \rangle are the arguments for that command. If you want \bibglssetlastgrouptitle to change the group title then you need to define it before the resource set. For example:

\begin{verbatim}
\newcommand*{\bibglssetlastgrouptitle}[2]{%
  \glsxtrsetgrouptitle{#1#2}{Foreign Words}%
}\GlsXtrLoadResources[src={entries}]
\end{verbatim}

If you need to change a particular group title, then it has to be done after the resource set:

\begin{verbatim}
\GlsXtrLoadResources[src={entries}]
\glsxtrsetgrouptitle 
  {\bibglsslettergroup\{Ø\}{Ø}\{214748112\}{}}% label 
  {Foreign Words}% title
\end{verbatim}

\bibglshypergroup

\begin{verbatim}
\bibglshypergroup{⟨type⟩}{⟨group-id⟩}
\end{verbatim}

If the .log file indicates that hyperref has been loaded and the --group switch is used, then this command will be used to create the navigation information for glossary styles such as indexhypergroup.

Top-Level Groups Only

The default group-level setting will only create groups for the top-level entries. Any sub-entries are considered to be part of the top-level entry’s group. If hierarchical groups are enabled, the commands defined in section 6.4 are provided.

\bibglssetlettergrouptitle

For each letter group that’s detected, bib2gls will write the line:

\begin{verbatim}
\bibglssetlettergrouptitle{⟨title⟩}{⟨letter⟩}{⟨id⟩}{⟨type⟩}
\end{verbatim}

in the .glstex file, which sets the group’s title using:
6.4 Letter Groups

\glsxtrsetgrouptitle\{\langle group label \rangle\}\{\langle group title \rangle\}

where the \langle group label \rangle part matches the corresponding group value.

Note that \bibglssetlettergrouptitle only has a single argument, but that argument contains the four arguments needed by \bibglslettergroup and \bibglslettergrouptitle. These arguments are as described above.

If \glsxtrsetgrouptitle has been defined (glossaries-extra version 1.14 onwards), then \bibglssetlettergrouptitle will be defined as:

\providecommand{\bibglssetlettergrouptitle}[1]{\glsxtrsetgrouptitle{\bibglslettergroup#1}{\bibglslettergrouptitle#1}}

If an earlier version of glossaries-extra is used, then this function can’t be supported and the command will be defined to simply ignore its argument. This will fall back on the original method of just using \langle title \rangle as the label.

Since \bibglssetlettergrouptitle is used in the .glstex file to set the group titles, the associated commands need to be defined before the resource file is loaded if their definitions require modification. After the resource file has been loaded, you can adjust the title of a specific group, but you’ll need to check the .glstex file for the appropriate arguments. For example, if the .glstex file contains:

\bibglssetlettergrouptitle\{\{Æ\{æ\{7274496\}\}\}\}

but you actually want the group title to appear as “Æ (AE)” instead of just “Æ”, then after the resource file has been loaded you can do:

\glsxtrsetgrouptitle\{\bibglslettergroup\{Æ\{æ\{7274496\}\}\}\}{Æ (AE)}% label
{Æ (AE)}% title

\bibglslettergroup

\bibglslettergrouptitle\{\langle title \rangle\}\{\langle letter \rangle\}\{\langle id \rangle\}\{\langle type \rangle\}

This command is used to determine the letter group label. The default definition is \langle type \rangle\langle id \rangle, which ensures that no problematic characters occur in the label since \langle type \rangle can’t contains special characters and \langle id \rangle is numeric. The \langle type \rangle is included in case there are multiple glossaries, since the hyperlink name must be unique.

\bibglslettergrouptitle

\bibglslettergrouptitle\{\langle title \rangle\}\{\langle letter \rangle\}\{\langle id \rangle\}\{\langle type \rangle\}

This command is used to determine the letter group title. The default definition is \unexpanded \{\langle title \rangle\}, which guards against any expansion issues that may arise with characters outside the basic Latin set.

For example:
6.4 Letter Groups

@entry{angstrom,
  name={\AA ngstr"om}
  description={a unit of length equal to one hundred-millionth
  of a centimetre}
}

The sort value is “Ångström”. With sort={en} the ⟨title⟩ part will be A but with sort={sv} the ⟨title⟩ part will be Å. In both cases the ⟨letter⟩ argument will be Å.

Take care if you are using a script that needs encapsulating. For example, with the CJKutf8 package the cjk characters need to be placed within the CJK environment, so any letter group titles that contain cjk characters will need special attention.

For example, suppose the .bib file contains entries in the form:

@dualentry{{label},
  name = {\cjkname{⟨cjk characters⟩}},
  description = {⟨English translation⟩}
}

and the document contains:

\usepackage{CJKutf8}
\usepackage[record,style={indexgroup},nomain]{glossaries-extra}
\newglossary*{japanese}{Japanese to English}
\newglossary*{english}{English to Japanese}
\newrobustcmd{\cjkname}[1]{\begin{CJK}{UTF8}{min}#1\end{CJK}}
\GlsXtrLoadResources[
  src={testcjk},% bib file
  sort={ja-JP},% locale used to sort primary entries
  dual-sort={en-GB},% locale used to sort dual entries
  type={japanese},% put the primary entry in the 'japanese' glossary
  dual-type={english},% put the dual entry in the 'english' glossary
  dual-prefix={en.}
]

then CJK characters will appear in the ⟨title⟩ argument of \bibglslettergrouptitle which causes a problem because they need to be encapsulated within the CJK environment. This can be more conveniently done with the user supplied \cjkname{⟨CJK characters⟩}, but the cjk characters need to be protected from expansion so \unexpanded is also needed. The new definition of \bibglslettergrouptitle needs to be defined before \GlsXtrLoadResources. For example:

\newcommand{\bibglslettergrouptitle}[4]{{\unexpanded{\cjkname{#1}}}}
There’s a slight problem here in that the English letter group titles also end up encapsulated. An alternative approach is to use the \textit{⟨type⟩} part to provide different forms. For example:

\begin{verbatim}
\newcommand*{\englishlettergroup}{\textit{⟨\text{character}⟩}\{⟨id⟩\}\{⟨type⟩\}}
\newcommand*{\japaneselettergroup}{\textit{⟨\text{character}⟩}\{⟨id⟩\}\{⟨type⟩\}}
\end{verbatim}

\texttt{\bibglslettergrouptitle}

The label and title for symbol groups are dealt with in a similar way to the letter groups, but in this case the title is set using:

\begin{verbatim}
\bibglssetothergrouptitle\{(character)\{id\}\{type\}\}
\end{verbatim}

This is defined in an analogous manner:

\begin{verbatim}
\providecommand{\bibglssetothergrouptitle}[1]{\\glsxtrsetgrouptitle{\bibglsothergroup#1}{\bibglsothergrouptitle#1}}
\end{verbatim}

where the group label is obtained using \bibglsothergroup and the group title is obtained from \bibglsothergrouptitle. Note that since non-alphabetic characters don’t have upper or lower case versions, there are only three arguments. The other difference between this and the letter group version is that the \textit{⟨id⟩} is given in hexadecimal format (corresponding to the character code).

For example, suppose my .bib file contains:

\begin{verbatim}
@entry{sauthor,
    name={/Author},
    description = {author string}
}
\end{verbatim}

If a locale sort is used, the leading slash / will be ignored and this entry will belong to the “A” letter group using the letter commands described above. If, instead, one of the character code sort methods are used, such as \textit{sort=⟨letter-case⟩}, then this entry will be identified as belonging to a symbol (or “other”) group and the title will be set using:

\begin{verbatim}
\bibglssetothergrouptitle\{/{\text{2F}}\}\{}
\end{verbatim}

\texttt{\bibglsothergroup}

\begin{verbatim}
\bibglsothergroup\{(character)\{id\}\{type\}\}
\end{verbatim}

This expands to the label for symbol groups. This just defaults to \texttt{\textit{glssymbols}} (ignoring all arguments), which replicates the label used when \texttt{makeindex} or \texttt{xindy} generate the glossary files.
6.4 Letter Groups

\bibglsothergrouptitle

\bibglsothergrouptitle\langle character\rangle\langle id\rangle\langle type\rangle

This expands to the title for symbol groups. This just expands to \glssymbolsgroupname by default.

\bibglsetemptygrouptitle

Used when the sort value degenerates to an empty string. This command sets the label and title.

\bibglsetemptygrouptitle\langle type\rangle

(Note the inner group, as with the other similar \bibglset...grouptitle commands.)

\bibglsemptrygroup

\bibglsemptrygroup\langle type\rangle

This expands to the label for empty groups. This defaults to glssymbols to make it consistent with non-letter groups (since the sort value likely contained unknown symbol commands).

\bibglsemptrygrouptitle

\bibglsemptrygrouptitle\langle type\rangle

This expands to the group title for empty group. This just expands to \glssymbolsgroupname by default.

\bibglsetnumbergrouptitle

The numeric sort methods (table 5.5) all create number groups instead of letter or symbol groups. These behave in an analogous way to the above.

\bibglsetnumbergrouptitle\langle value\rangle\langle id\rangle\langle type\rangle

In this case \langle value\rangle is the actual numeric sort value, and \langle id\rangle is a decimal number obtained from converting \langle value\rangle to an integer. This command is defined as:

\providecommand{\bibglsetnumbergrouptitle}[1]{% 
  \glsxtrsetgrouptitle{\bibglslnumbergroup#1}{\bibglslnumbergrouptitle#1}}

\bibglslnumbergroup

The number group label is obtained from:

\bibglslnumbergroup\langle value\rangle\langle id\rangle\langle type\rangle

This just defaults to glsnumbers.
6.4 Letter Groups

\bibglsnumbergrouptitle
The number group title is obtained from:
\bibglsnumbergrouptitle\{\langle\text{value}\rangle\}\{\langle\text{id}\rangle\}\{\langle\text{type}\rangle\}
This just defaults to \textit{\texttt{glsnumbersgroupname}}.

\bibglssetdatetimegrouptitle
The date-time sort methods (table 5.6) create date-time groups. These behave in an analogous way to the above.
\bibglssetdatetimegrouptitle\{\langle\text{YYYY}\rangle\}\{\langle\text{MM}\rangle\}\{\langle\text{DD}\rangle\}\{\langle\text{hh}\rangle\}\{\langle\text{mm}\rangle\}\{\langle\text{ss}\rangle\}\{\langle\text{zone}\rangle\}\{\langle\text{title}\rangle\}\{\langle\text{group-id}\rangle\}\{\langle\text{type}\rangle\}\}
This command is defined as:
\providecommand{\bibglssetdatetimegrouptitle}[1]{%\glsxtrsetgrouptitle{\bibglsdatetimengroup#1}{}{\bibglssetdatetimegrouptitle#1}}
\bibglsdatetimengroup
\bibglsdatetimengroup\{\langle\text{YYYY}\rangle\}\{\langle\text{MM}\rangle\}\{\langle\text{DD}\rangle\}\{\langle\text{hh}\rangle\}\{\langle\text{mm}\rangle\}\{\langle\text{ss}\rangle\}\{\langle\text{zone}\rangle\}\{\langle\text{title}\rangle\}\{\langle\text{group-id}\rangle\}\{\langle\text{type}\rangle\}\}
This command is used for date-time group labels with \texttt{datetime} sorting (table 5.6). This has ten arguments, which means a little trickery is needed to deal with the tenth argument. The default definition is:
\providecommand{\bibglsdatetimengroup}[9]{\#1\#2\#3\@firstofone}
This forms the group label from the year \langle\text{YYYY}\rangle, month \langle\text{MM}\rangle, day \langle\text{DD}\rangle and \langle\text{type}\rangle.

\bibglsdatetimengrouptitle
\bibglsdatetimengrouptitle\{\langle\text{YYYY}\rangle\}\{\langle\text{MM}\rangle\}\{\langle\text{DD}\rangle\}\{\langle\text{hh}\rangle\}\{\langle\text{mm}\rangle\}\{\langle\text{ss}\rangle\}\{\langle\text{zone}\rangle\}\{\langle\text{title}\rangle\}\{\langle\text{group-id}\rangle\}\{\langle\text{type}\rangle\}\}
This command is used for date-time group titles with \texttt{datetime} sorting (table 5.6). The default definition is:
\providecommand{\bibglsdatetimengrouptitle}[9]{\#1-\#2-\#3\@gobble}
This sets the title to the numeric \langle\text{YYYY}\rangle-\langle\text{MM}\rangle-\langle\text{DD}\rangle but may be redefined as appropriate.
\bibglsssetdategrouptitle

The date sort methods (table 5.6) create date groups (the time isn’t included). These behave in an analogous way to the above.

\bibglsssetdategrouptitle{\{\langle YYYY\rangle\}\{\langle MM\rangle\}\{\langle DD\rangle\}\{\langle G\rangle\}\{\langle title\rangle\}\{\langle group-id\rangle\}\{\langle type\rangle\}\}}

This command is defined as:

\providecommand{\bibglsssetdategrouptitle}[1]{{% 
\glsxtrsetgrouptitle{\bibglssdategroup#1}{\bibglssdategrouptitle#1}}}

\bibglssdategroup

\bibglssdategroup{\{\langle YYYY\rangle\}\{\langle MM\rangle\}\{\langle DD\rangle\}\{\langle G\rangle\}\{\langle title\rangle\}\{\langle group-id\rangle\}\{\langle type\rangle\}}

This command is used for date group labels with date (no time) sorting (table 5.6). The default definition is:

\providecommand{\bibglssdategroup}[7]{\#1#2#4#7}

This forms the group label from the year, month, era and type. In this case, the era is a textual representation not the numeric value used in calculating the sort value.

\bibglssdategrouptitle

\bibglssdategrouptitle{\{\langle YYYY\rangle\}\{\langle MM\rangle\}\{\langle DD\rangle\}\{\langle G\rangle\}\{\langle title\rangle\}\{\langle group-id\rangle\}\{\langle type\rangle\}}

This command is used for date group titles with date (no time) sorting (table 5.6). The default definition is:

\providecommand{\bibglssdategrouptitle}[7]{\#1\#2}

This just sets the title to the numeric year-month form \langle YYYY\rangle-\langle MM\rangle.

\bibglsssettimegrouptitle

The time sort methods (table 5.6) create time groups (the date isn’t included). These behave in an analogous way to the above.

\bibglsssettimegrouptitle{\{\langle hh\rangle\}\{\langle mm\rangle\}\{\langle ss\rangle\}\{\langle zone\rangle\}\{\langle title\rangle\}\{\langle group-id\rangle\}\{\langle type\rangle\}}

This command is defined as:

\providecommand{\bibglsssettimegrouptitle}[1]{{% 
\glsxtrsetgrouptitle{\bibglstimegroup#1}{\bibglstimegrouptitle#1}}

351
\bibglstimegroup

This command is used for time group labels with time (no date) sorting (table 5.6). This command is defined as:

\providecommand{\bibglstimegroup}[7]{#1#2#7}

\bibglstimegrouptitle

This command is used for time group titles with time (no date) sorting (table 5.6). This command is defined as:

\providecommand{\bibglstimegrouptitle}[7]{#1}

\bibglssetunicodegrouptitle

This command is used to assign the group titles when the group formation is set to any value other than the default. For example, this command will be used with group-formation= {codepoint}. The label is obtained from \bibglsunicodegroup and the title is obtained from \bibglsunicodegrouptitle.

\bibglsunicodegroup

The ⟨label⟩ depends on the group-formation setting:

- group-formation={codepoint}: the ⟨label⟩ is the Unicode value of ⟨character⟩ (converted to lower case and decomposed, if applicable);
- group-formation={unicode category}: the ⟨label⟩ is the Unicode category of ⟨character⟩ (for example, Lu means an upper case letter);
- group-formation={unicode script}: the ⟨label⟩ is the Unicode script associated with ⟨character⟩ (for example, LATIN);
- group-formation={unicode category and script}: the ⟨label⟩ identifies both the Unicode category and script associated with ⟨character⟩ (for example, Lu.LATIN).
(Similarly for secondary-group-formation and dual-group-formation.) By default this command expands to \(\text{(type)}\text{(label)}\).

The \(\text{(character)}\) is the first significant character of the sort value. The \(\text{(id)}\) is the hexadecimal code of (possibly decomposed) \(\text{(character)}\). The case of codepoint \(\text{(id)}\) may or may not correspond to the case of \(\text{(character)}\).

For example, with group-formation=\{codepoint\}, an unset type and a sort value of “Ångström” with “Å” as a significant character distinct from “A” then the group field will be assigned using:

\[
group=\{\text{bibglsunicodegroup}{â}{Å}{C5}\}\}
\]

whereas with group-formation=\{unicode category and script\} it will be:

\[
group=\{\text{bibglsunicodegroup}{Lu.LATIN}{Å}{C5}\}\}
\]

(upper case Latin letter).

If instead “Å” is considered equivalent to “A” according to the collator, then with group-formation=\{codepoint\}, the value will be:

\[
group=\{\text{bibglsunicodegroup}{a}{Å}{61}\}\}
\]

Note that the \(\text{(id)}\) is now 0x61 (the decomposed “A” converted to lower case) not 0xC5.

\textbf{\textbackslash bibglsunicodegrouptitle}\{\langle label\rangle\}\{\langle character\rangle\}\{\langle id\rangle\}\{\langle type\rangle\}

The title for Unicode group formations is simply defined as \textbf{\unexpanded{\langle label\rangle}} so you will need to change it to something more appropriate. For example (before the resource set):

\[
\textbf{\newcommand}{\textbf{\bibglsunicodegrouptitle}}{[4]\%\}
\]

\[
\textbf{\ifnum"#3>64}
\textbf{\ifnum"#3 < 91}
\textbf{A--Z\%}
\textbf{\else}
\textbf{\ifnum"#3 > 96}
\textbf{\ifnum"#3 < 123}
\textbf{A--Z\%}
\textbf{\fi}
\textbf{\fi}
\textbf{\fi}
\textbf{\fi}
\]

This will make the title “A–Z” if \(\text{id}\) is greater than 64 and less than 91 or greater than 96 and less than 123 (and will be empty otherwise).

Note that this setting can create an odd effect if the sorting causes the groups to be split up. For example, if some of the sort values start with extended or non-Latin characters this can break up the groups. First check how the group labels are assigned using:
6.4 Letter Groups

\newcommand{\bibglsunicodegrouptitle}{\bibglsunicodegroup}

then adjust the definition of \bibglsunicodegroup until the grouping is correct, and then change the definition of \bibglsunicodegrouptitle so that the title is correct.

\bibglssetmergedgrouptitle

Used when groups are merged by merge-small-groups. This command sets the label and title.

\bibglssetmergedgrouptitle{\{id\}\{type\}\{n\}\{g_1\}...\{g_{n-1}\}\{g_n\}}

(Note the inner group, as with the other similar \bibglsset...grouptitle commands.)

\bibglsmergedgroup

\bibglsmergedgroup{(id)\{type\}\{n\}\{g_1\}...\{g_{n-1}\}\{g_n\}}

This expands to the label for merged groups.

\bibglsmergedgrouptitle

\bibglsmergedgrouptitle{(id)\{type\}\{n\}\{g_1\}...\{g_{n-1}\}\{g_n\}}

This expands to the group title for merged groups.

\bibglsmergedgroupfmt

\bibglsmergedgroupfmt{(n)\{g_1\}...\{g_{n-1}\}\{g_n\}}

Used by \bibglsmergedgrouptitle and \bibglsmergedgrouphierfmt to format merged group titles. The first argument \(n\) is the total number of groups that have been merged, the second argument, \(g_1\) is the first group title, \(\{g_2\}...\{g_n\}\) are the middle group titles (empty if \(n = 2\)), and \(g_n\) is the last group title.

The default definition is:

\providecommand{\bibglsmergedgroupfmt}[4]{#2, \ifcase#1\or\or\or #3, \else... \fi #4}

Hierarchical Groups

Hierarchical letter groups are set with analogous commands that have hier appended to the command name. There are also two extra arguments, \(\langle parent\rangle\) (the parent entry’s label) and \(\langle level\rangle\) (the hierarchical level).
\bibglsgrouplevel

This command is used when group-level is used to apply group formations for different hierarchical levels. The \langle label \rangle argument is the label that would normally be applied to level 0. The \langle n \rangle argument is the hierarchical level (0 for top-level entries). By default, this command simply expands to \langle label \rangle.

\bibglshiersubgrouptitle

Hierarchical group titles are formatted using:

\bibglshiersubgrouptitle{level}{parent}{title}

This is defined as:

\providecommand{\bibglshiersubgrouptitle}[3]{\ifnum#1>0 \Glsxtrhiername{#2} / \fi #3}

The first argument \langle level \rangle is the hierarchical level, the second argument is the parent entry label (empty for \langle level \rangle = 0), and the third argument \langle title \rangle is the normal title that would apply to the group with the default group-level={0} setting. Note that this uses \Glsxtrhiername{\langle parent \rangle} if \langle level \rangle > 0. If this isn’t required then redefine \bibglshiersubgrouptitle to simply expand to \langle title \rangle (#3).

\bibglssetlettergrouptitlehier

As \bibglssetlettergrouptitle but used for hierarchical groups.

\bibglslettergrouphier

As \bibglslettergroup but used for hierarchical groups. It expands to the hierarchical letter group label. This is defined as:

\providecommand{\bibglslettergrouphier}[6]{#4#5#3}
6.4 Letter Groups

\bibglslettergrouptitlehier
\bibglslettergrouptitlehier{⟨title⟩}{⟨letter⟩}{⟨id⟩}{⟨type⟩}{⟨parent⟩}{⟨level⟩}
As \bibglslettergrouptitle but used for hierarchical groups. It expands to the hierarchical letter group title. This is defined as:
\providecommand{\bibglslettergrouptitlehier}[6]{\protect\bibglshiersubgrouptitle{#6}{#5}{\unexpanded{#1}}}

\bibglssetothergrouptitlehier
\bibglssetothergrouptitlehier{⟨character⟩}{⟨id⟩}{⟨type⟩}{⟨parent⟩}{⟨level⟩}
As \bibglssetothergrouptitle but used for hierarchical groups.

\bibglsothergrouphier
\bibglsothergrouphier{⟨character⟩}{⟨id⟩}{⟨type⟩}{⟨parent⟩}{⟨level⟩}
As \bibglsothergroup but used for hierarchical groups. It expands to the hierarchical other group label. This is defined as:
\providecommand{\bibglsothergrouphier}[5]{#3#4\glssymbols}

\bibglsothergrouptitlehier
\bibglsothergrouptitlehier{⟨character⟩}{⟨id⟩}{⟨type⟩}{⟨parent⟩}{⟨level⟩}
As \bibglsothergrouptitle but used for hierarchical groups. It expands to the hierarchical other group title. This is defined as:
\providecommand{\bibglsothergrouptitlehier}[5]{\protect\bibglshiersubgrouptitle{#5}{#4}{\protect\glssymbolsgroupname}}

\bibglssetemptygrouptitlehier
\bibglssetemptygrouptitlehier{⟨type⟩}{⟨parent⟩}{⟨level⟩}
As \bibglssetemptygrouptitle but used for hierarchical groups.

\bibglsemptygrouphier
\bibglsemptygrouphier{⟨type⟩}{⟨parent⟩}{⟨level⟩}
As \bibglsemptygroup but used for hierarchical groups. It expands to the hierarchical empty group label. This is defined as:
\providecommand{\bibglsemptygrouphier}[3]{#1#2\glssymbols}
6.4 Letter Groups

\bibglsemptygrouptitlehier

\bibglsemptygrouptitlehier{⟨\textit{type}⟩}{⟨\textit{parent}⟩}{⟨\textit{level}⟩}

As \texttt{\bibglsemptygrouptitlehier} but used for hierarchical groups. It expands to the hierarchical empty group title. This is defined as:

\providecommand{\bibglsemptygrouptitlehier}{\protect\bibglshiersubgrouptitle{#3}{#2}{\protect\glssymbolsgroupname}}

\bibglssetnumbergrouptitlehier

\bibglssetnumbergrouptitlehier{⟨\textit{value}⟩}{⟨\textit{id}⟩}{⟨\textit{type}⟩}{⟨\textit{parent}⟩}{⟨\textit{level}⟩}

As \texttt{\bibglssetnumbergrouptitlehier} but used for hierarchical groups.

\bibglshnumbergrouphier

\bibglshnumbergrouphier{⟨\textit{value}⟩}{⟨\textit{id}⟩}{⟨\textit{type}⟩}{⟨\textit{parent}⟩}{⟨\textit{level}⟩}

As \texttt{\bibglshnumbergrouphier} but used for hierarchical groups. It expands to the hierarchical number group label. This is defined as:

\providecommand{\bibglshnumbergrouphier}{\protect\bibglshiersubgrouptitle{#3}{#4}{\protect\glsnumbersgroupname}}

\bibglsetnumbergrouptitlehier

\bibglsetnumbergrouptitlehier{⟨\textit{value}⟩}{⟨\textit{id}⟩}{⟨\textit{type}⟩}{⟨\textit{parent}⟩}{⟨\textit{level}⟩}

As \texttt{\bibglsetnumbergrouptitlehier} but used for hierarchical groups. It expands to the hierarchical number group title. This is defined as:

\providecommand{\bibglsetnumbergrouptitlehier}{\protect\bibglshiersubgrouptitle{#5}{#4}{\protect\glsnumbersgroupname}}

\bibglsetdatetimegrouptitlehier

\bibglsetdatetimegrouptitlehier{⟨\textit{YYYY}⟩}{⟨\textit{MM}⟩}{⟨\textit{DD}⟩}{⟨\textit{hh}⟩}{⟨\textit{mm}⟩}{⟨\textit{ss}⟩}{⟨\textit{zone}⟩}{⟨\textit{title}⟩}{⟨\textit{group-id}⟩}{⟨\textit{type}⟩}{⟨\textit{parent}⟩}{⟨\textit{level}⟩}

As \texttt{\bibglsetdatetimegrouptitlehier} but used for hierarchical groups.
As \bibglsdatetimegrouphier but used for hierarchical groups. It expands to the hierarchical date-time group label. Since this has more than 9 arguments, it is defined in two parts:

\providecommand{\bibglsdatetimegrouphier}{\bibglsdatetimegroup-hierfinalargs}

The final arguments are obtained with \bibglsdatetimegrouphierfinalargs.

\bibglsdatetimegrouphierfinalargs

\providecommand*{\bibglsdatetimegrouphierfinalargs}{\bibglshiersubgrouptitle{#3}{#1}}

In this case, the first argument is set by \bibglsdatetimegrouptitlehier to the date \((YYYY)-(MM)-(DD))\). The remaining arguments are the \(type\), \(parent\) and \(level\) information.
\bibglssetdategrouptitlehier
\bibglssetdategrouptitlehier{{\langle YYYY \rangle}{\langle MM \rangle}{\langle DD \rangle}{\langle G \rangle}{\langle title \rangle}{\langle group-id \rangle}{\langle type \rangle}{\langle parent \rangle}{\langle level \rangle}}

As $\bibglssetdategrouptitle$ but used for hierarchical groups.

\bibglsdategrouphier
\bibglsdategrouphier{{\langle YYYY \rangle}{\langle MM \rangle}{\langle DD \rangle}{\langle G \rangle}{\langle title \rangle}{\langle group-id \rangle}{\langle type \rangle}{\langle parent \rangle}{\langle level \rangle}}

As $\bibglsdategrouptitle$ but used for hierarchical groups. It expands to the hierarchical date group label. This is defined as:
\providecommand{$\bibglsdategrouptitlehier$}[9]{#1#2#4#7#8}

\bibglsdategrouptitlehier
\bibglsdategrouptitlehier{{\langle YYYY \rangle}{\langle MM \rangle}{\langle DD \rangle}{\langle G \rangle}{\langle title \rangle}{\langle group-id \rangle}{\langle type \rangle}{\langle parent \rangle}{\langle level \rangle}}

As $\bibglsdategrouptitle$ but used for hierarchical groups. It expands to the hierarchical date group title. This is defined as:
\providecommand{$\bibglsdategrouptitlehier$}[9]{\protect\bibglshiersubgrouptitle{#9}{#8}{#1-#2}}

\bibglssettimegrouptitlehier
\bibglssettimegrouptitlehier{{\langle hh \rangle}{\langle mm \rangle}{\langle ss \rangle}{\langle zone \rangle}{\langle title \rangle}{\langle group-id \rangle}{\langle type \rangle}{\langle parent \rangle}{\langle level \rangle}}

As $\bibglssettimegrouptitle$ but used for hierarchical groups.

\bibglstimegrouphier
\bibglstimegrouphier{{\langle hh \rangle}{\langle mm \rangle}{\langle ss \rangle}{\langle zone \rangle}{\langle title \rangle}{\langle group-id \rangle}{\langle type \rangle}{\langle parent \rangle}{\langle level \rangle}}

As $\bibglstimegroup$ but used for hierarchical groups. It expands to the hierarchical time group label. This is defined as:
\providecommand{$\bibglstimegrouphier$}[9]{#1#2#7#8}
\bibglstimegrouptitlehier
\bibglstimegrouptitlehier{\langle hh \rangle \langle mm \rangle \langle ss \rangle \langle zone \rangle \langle title \rangle \langle group-id \rangle \\
\langle type \rangle \langle parent \rangle \langle level \rangle}

As \texttt{\bibglstimegrouptitle} but used for hierarchical groups. It expands to the hierarchical time group title. This is defined as:
\providecommand{\bibglstimegrouptitlehier}[9]{\protect\bibglshiersubgrouptitle{#9}{#8}{#1}}

\bibglsset_unicode_grouptitlehier
\bibglsset_unicode_grouptitlehier{\langle label \rangle \langle character \rangle \langle id \rangle \langle type \rangle \\
\langle parent \rangle \langle level \rangle}}

As \texttt{\bibglsset_unicode_grouptitle} but used for hierarchical groups.

\bibglunicodegrouphier
\bibglunicodegrouphier{\langle label \rangle \langle character \rangle \langle id \rangle \langle type \rangle \\
\langle parent \rangle \langle level \rangle}}

As \texttt{\bibglunicodegroup} but used for hierarchical groups. This expands to the hierarchical group label:
\providecommand{\bibglunicodegrouphier}[6]{#4#5#3}

\bibglunicodegrouptitlehier
\bibglunicodegrouptitlehier{\langle label \rangle \langle character \rangle \langle id \rangle \langle type \rangle \langle parent \rangle \\
\langle level \rangle}}

As \texttt{\bibglunicodegrouptitle} but used for hierarchical groups. This expands to the hierarchical group title:
\providecommand{\bibglunicodegrouptitlehier}[6]{\protect\bibglshiersubgrouptitle{#6}{#5}{\unexpanded{#1}}}
6.4 Letter Groups

\bibglsmergedgrouphier

\bibglsmergedgrouphier{{id}}{{type}}{{n}}{{g_1}}{{g_2}...{{g}_{n-1}}}{{g_n}}{{parent}}{{level}}

As \bibglsmergedgroup but used for hierarchical groups. This expands to the hierarchical group label:

\providecommand{\bibglsmergedgrouphier}{merged.#1}

\bibglsmergedgrouptitlehier

\bibglsmergedgrouptitlehier{{id}}{{type}}{{n}}{{g_1}}{{g_2}...{{g}_{n-1}}}{{g_n}}{{parent}}{{level}}

As \bibglsmergedgrouptitle but used for hierarchical groups. This expands to the hierarchical group title:

\providecommand{\bibglsmergedgrouptitlehier}{\ifnum#8=0\bibglsmergedgroupfmt{#3}{#4}{#5}{#6}\else\bibglsmergedgrouphierfmt{#3}{#4}{#5}{#6}\fi}

This uses \bibglsmergedgroupfmt ((level) = 0) or \bibglsmergedgrouphierfmt ((level) > 0) to format the title.

\bibglsmergedgrouphierfmt

\bibglsmergedgrouphierfmt{{n}}{{g_1}}{{g_2}...{{g}_{n-1}}}{{g_n}}

Used by \bibglsmergedgrouphierfmt to format merged hierarchical group titles. The first argument \langle n\rangle is the total number of groups that have been merged, the second argument, \langle g_1\rangle is the first group title, \langle g_2⟩...⟨g_n⟩ are the middle group titles (empty if \langle n\rangle = 2), and \langle g_n\rangle is the last group title.

The default definition depends on whether or not bib2gls has detected hyperref in the document’s .log file. If it has, then the definition is:

\providecommand{\bibglsmergedgrouphierfmt}{#2,\textorpdfstring{{\def\bibglshiersubgrupptitle##1##2##3##3\ifcase#1\or\or\or#3,\else\ldots,\fi#4}}{\ifcase#1\or\or\or#3,\else\ldots,\fi#4}}\{\ifcase#1\or\or\or#3,\else\ldots,\fi#4\}

Otherwise the definition is:
6.5 Flattened Entries

These commands relate to the way the name field is altered when flattening lonely child entries with the flatten-lonely option.

\bibglsflattenedhomograph

\bibglsflattenedhomograph{⟨name⟩}{⟨parent label⟩}

The default definition simply does ⟨name⟩. This command is used if the child and parent names are identical. For example, suppose the .bib file contains:

@index{super.glossary, name={glossary}}

@entry{glossarycol,
    parent={super.glossary},
    description={collection of glosses}
}

@entry{glossarylist,
    parent={super.glossary},
    description={list of technical words}
}

The child entries don’t have a name field, so the value is assumed to be the same as the parent’s name field. Here’s an example document where both child entries are used:

\documentclass{article}

\usepackage[record,subentrycounter,style={treenoname}]{glossaries-extra}

\GlsXtrLoadResources[src={entries}]{glossaries-extra}

\begin{document}
\gls{glossarycol} (collection) vs \gls{glossarylist} (list).
\end{document}
This uses one of the glossary styles designed for homographs and the glossary has the structure:

\begin{description}
\item[glossary] 1) collection of glosses 1
\item[glossary] 2) list of technical words 1
\end{description}

If only one child entry is selected, then the result looks a little odd. For example:

\begin{description}
\item[glossary] 1) collection of glosses 1
\end{description}

With the flatten-lonely option, the parent is removed and the child is moved up a hierarchical level. With flatten-lonely={postsort} this would normally adjust the name so that it appears as \textit{(parent name), (child name)} but in this case it would look a little odd for the name to appear as “glossary, glossary” so instead the name is set to:

\begin{verbatim}
\bibglsflattenedhomograph{glossary}{super.glossary}
\end{verbatim}

(where the first argument is the original name and the second argument is the label of the parent entry).

This means that the name simply appears as “glossary”, even if the flatten-lonely={postsort} option is used. Note that if the parent entry is removed, the parent label won’t be of much use. You can test for existence using \ifglsentryexists in case you want to vary the way the name is displayed according to whether or not the parent is still present.

\begin{verbatim}
\bibglsflattenedchildpresort
\end{verbatim}

Used by the flatten-lonely={presort} option. This defaults to just \textit{(child name)}. If you want to change this, remember that you can let the interpreter know by adding the definition to @preamble. For example:

@preamble{"\providecommand{\bibglsflattenedchildpresort}[2]{#1 (#2)}"}

\begin{verbatim}
\bibglsflattenedchildpostsort
\end{verbatim}

Used by the flatten-lonely={postsort} option. This defaults to \textit{(parent name)}, \textit{(child name)}.
Note that the arguments are in the reverse order to those of the previous command. This is done to assist the automated first letter upper-casing. If either command is redefined to alter the ordering, then this can confuse the case-changing mechanism, in which case you may want to consider switching on the expansion of the name field using:

\glssetexpandfield{name}

(before \GlsXtrLoadResources).

\section*{6.6 Other}

\listglscoptoglossary

This command is provided if the copy-to-glossary option is set and is used to copy an entry to another glossary. The definition is:

\providecommand{\listglscoptoglossary}[2]{%\ifglossaryexists*{}%{\GlsXtrIfInGlossary{#1}{#2}{}}%}

This ensures that the entry is only copied if the glossary exists and if the entry hasn’t already been copied to it.

This command isn’t used by the action={copy} or action={copy or define} settings, which use \glsxtrcopytoglossary directly.

\listglssettotalrecordcount

This command is provided if --record-count is used. It’s used to set the recordcount field to the total number of records for the given entry. This is defined as:

\providecommand*{\listglssettotalrecordcount}[3]{%\GlsXtrSetField{#1}{recordcount}{#2}{#3}{}}

\listglssetrecordcount

This command is provided if --record-count is used. It’s used to set the recordcount. \langle counter \rangle field to the total number of records associated with the given counter for the given entry. This is defined as:
This command is provided if --record-count-unit is used. It's used to set the \recordcount, \counter, \location field to the total number of records associated with the given location for the given entry. This is defined as:

```latex
\providecommand*{\bibglssetlocationrecordcount}{3}{%
\GlsXtrSetField{#1}{recordcount.#2}{#3}%
}\bibglssetlocationrecordcount{⟨entry-label⟩}{⟨counter⟩}{⟨location⟩}{⟨value⟩}
```

Used by the back link options, this just defaults to:

```latex
\bibglshyperlink{⟨text⟩}{⟨label⟩}
```

This is used by set-widest to set the widest name for the given hierarchical level where the glossary type can't be determined. This is defined as:

```latex
\providecommand*{\bibglssetwidest}{2}{%
\GlsxtrSetField{#1}{widest.#2}{#3}%
}\bibglssetwidest{⟨level⟩}{⟨name⟩}
```

Since this isn’t scoped, this will affect other glossaries. In general, if you have more than one glossary it’s best to set the type using options like type.
6.6 Other

\bibglsetwidestfortype

\bibglsetwidestfortype{⟨type⟩}{⟨level⟩}{⟨name⟩}

This is used by set-widest to set the widest name for the given hierarchical level where the glossary type is known. This is defined as:

\providecommand*{\\bibglsetwidestfortype}{[3]{% \glsxtrSetWidest{#1}{#2}{#3} %}}

if \glsxtrSetWidest has been defined, or:

\providecommand*{\\bibglsetwidestfortype}{[3]{% \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}} %}}

if \glsupdatewidest is defined, otherwise it will be defined to use \bibglsetwidest:

\providecommand*{\\bibglsetwidestfortype}{[3]{% \apptoglossarypreamble[#1]{\bibglsetwidest[#2]{#3}} %}}

Since the glossary preamble is scoped, this won’t affect other glossaries.

\bibglsetwidestfallback

\bibglsetwidestfallback{⟨glossary list⟩}

This is used by set-widest instead of \bibglsetwidest when all name fields end up as an empty string when interpreted by bib2gls. This typically means that all the name fields contain unknown commands. This fallback command will use:

\glsxtrSetWidestFallback{2}{⟨glossary list⟩}

if defined otherwise it will use \glsFindWidestLevelTwo, which sets the widest name for the top-level and first two sub-levels across all the listed glossaries.

\bibglsetwidestfortypefallback

\bibglsetwidestfortypefallback{⟨type⟩}

This is used by set-widest instead of \bibglsetwidestfortype when all name fields end up as an empty string when interpreted by bib2gls. This typically means that all the name fields contain unknown commands. This fallback command will append \bibglsetwidestfallback to the glossary preamble for the given type.
This is used by \bibglssetwidest instead of \bibglssetwidest when all name fields end up as an empty string when interpreted by bib2gls. This typically means that all the name fields contain unknown commands. This fallback command will use:
\glsxtrSetWidestFallback{0}{⟨glossary list⟩}
if defined otherwise it will use \glsFindWidestTopLevelName, which sets the widest name for the top-level.

This is used by \bibglssetwidestfortype instead of \bibglssetwidestfortype when all name fields end up as an empty string when interpreted by bib2gls. This typically means that all the name fields contain unknown commands. This fallback command will append \bibglssetwidesttoplevelfallback to the glossary preamble of the given type.

This is used when \bibtex-contributor-fields is set. The definition depends on whether or not \DTLformatlist has been defined:

\ifdef\DTLformatlist
{% datatool v2.28+
 \providecommand*{\bibglscontributorlist}[2]{\DTLformatlist{#1}}
}
{% datatool v2.27 or earlier
 \providecommand*{\bibglscontributorlist}[2]{%
 \def\bibgls@sep{}
 \@for\bibgls@item:=#1\do{\bibgls@sep\bibgls@item\def\bibgls@sep{, }}%
}
}

The second argument allows you to provide definitions like:

\newcommand*{\bibglscontributorlist}[2]{%
 \ifcase#2
 \or
6.6 Other

name: \else
names: \fi
\DTLformatlist{#1}\

\bibglscontributor

\bibglscontributor{(forenames)\{von-part\}\{surname\}\{suffix\}

This is used when \texttt{bibtex-contributor-fields} is set. The definition depends on the value of \texttt{contributor-order}. Note that if you have multiple resource sets, that option governs the way \texttt{bib2gls}'s version of \texttt{\bibglscontributor} behaves. The definition is written to the \texttt{.glstex} using \texttt{\providecommand}, so \LaTeX{} will only pick up the first definition.

\bibglsdatetime

\bibglsdatetime{(year)\{month\}\{day-of-month\}\{day-of-week\}\{day-of-year\}\{era\}\{hour\}\{minute\}\{second\}\{millisec\}\{dst\}\{zone\}\{original\}

Used to encapsulate date-time fields identified with \texttt{date-time-fields}. Since \texttt{\bibglsdatetime} requires more than nine arguments, the remaining four arguments are picked up with:
\bibglsdatetimeremainder{\{millisec\}\{dst\}\{zone\}\{original\}

The default definitions are:
\providecommand{\bibglsdatetime}[9]{\bibglsdatetimeremainder}
\providecommand{\bibglsdatetimeremainder}[4]{#4}

\bibglsdate

\bibglsdate{(year)\{month\}\{day-of-month\}\{day-of-week\}\{day-of-year\}\{era\}\{original\}

Used to encapsulate date fields identified with \texttt{date-fields}. The default definition is:
\providecommand{\bibglsdate}[7]{#7}

\bibglstime

\bibglstime{(hour)\{minute\}\{second\}\{millisec\}\{dst\}\{zone\}\{original\}

Used to encapsulate date fields identified with \texttt{time-fields}. The default definition is:
\providecommand{\bibglstime}[7]{#7}
\bibglprimaryprefixlabel

\bibglprimaryprefixlabel\{\langle prefix \rangle\}

A hook to pick up the primary prefix label (identified with label-prefix) if required. This does nothing by default. If required, this command should be defined before the resource set is loaded.

\bibglprimaryprefixlabel

\bibglprimaryprefixlabel\{\langle prefix \rangle\}

A hook to pick up the dual prefix label (identified with dual-prefix) if required. This does nothing by default. If required, this command should be defined before the resource set is loaded.

\bibglprimaryprefixlabel

\bibglprimaryprefixlabel\{\langle prefix \rangle\}

A hook to pick up the tertiary prefix label (identified with tertiary-prefix) if required. This does nothing by default. If required, this command should be defined before the resource set is loaded.

\bibglprimaryprefixlabel

\bibglprimaryprefixlabel\{(n)\}\{\langle prefix \rangle\}

A hook to pick up the \langle n\rangle th external prefix label (identified with ext-prefixes) if required. This does nothing by default and won’t be used if the list of external prefixes is empty. If required, this command should be defined before the resource set is loaded.

\bibglprimaryprefixlabel

\bibglprimaryprefixlabel

Expands to a literal hash character (#).

\bibglprimaryprefixlabel

\bibglprimaryprefixlabel

Expands to a literal underscore character ( _ ).
\bibglsdollarchar

Expands to a literal dollar character ($). 

\bibglsampersandchar

Expands to a literal ampersand character (&). 

\bibglscircumchar

Expands to a literal circumflex character (^). 

\bibglsaposchar

Expands to a literal apostrophe character ('). This command is only provided if --replace-quotes is used. 

\bibglsdoublequotechar

Expands to a literal double-quote character ("). This command is only provided if --replace-quotes is used. 

\bibglssupercase

\bibglssupercase{⟨text⟩}

Converts ⟨text⟩ to upper case. This just uses \glsupercase (if glossaries v4.50+, glossaries-extra v1.49+ and mfirstuc v2.08+) or \MakeTextUppercase by default. 

\bibglslowercase

\bibglslowercase{⟨text⟩}

Converts ⟨text⟩ to lower case. This just uses \glslowercase (if glossaries v4.50+, glossaries-extra v1.49+ and mfirstuc v2.08+) \MakeTextLowercase by default.
\texttt{\bibglstitlecase}
\begin{verbatim}
\bibglstitlecase{(text)}
\end{verbatim}
Converts \textit{(text)} to title case. This just uses \texttt{\capitalisewords} by default.

\texttt{\bibglsfirstuc}
\begin{verbatim}
\bibglsfirstuc{(text)}
\end{verbatim}
Converts the first letter of \textit{(text)} to upper case. This just uses \texttt{\makefirstuc} by default.

\texttt{\bibglsdefinitionindex}
\begin{verbatim}
\bibglsdefinitionindex{(label)}
\end{verbatim}
If \texttt{\save-definition-index} has been set this command expands to the definition index of the entry identified by \textit{(label)}. This command will only be provided in the \texttt{.glstex} file if \texttt{\save-definition-index} has been set. However, the command is always defined by the \TeX\ parser library but will expand to empty if the associated resource option hasn't been set.

\texttt{\bibglsuseindex}
\begin{verbatim}
\bibglsuseindex{(label)}
\end{verbatim}
If \texttt{\save-use-index} has been set this command expands to the order of use index of the entry identified by \textit{(label)}. This command will only be provided in the \texttt{.glstex} file if \texttt{\save-use-index} has been set. However, the command is always defined by the \TeX\ parser library but will expand to empty if the associated resource option hasn't been set.
7 Converting Existing .tex to .bib

If you have already been using the glossaries or glossaries-extra package with a large file containing all your definitions using commands like \newglossaryentry, then you can use the supplementary tool convertgls2bib to convert the definitions to the .bib format required by bib2gls. The syntax is:

convertgls2bib [(options)] (tex file) (bib file)

where (tex file) is the .tex file and (bib file) is the .bib file. This application is less secure than bib2gls as it doesn’t use kpsewhich to check openin_any and openout_any. Take care not to accidentally overwrite existing .bib files as there’s no check to determine if (bib file) already exists with the default --overwrite.

If the .bib extension is missing from (bib file), it will be added. The extension is required for (tex file).

7.1 Command Line Arguments

The (options) recognised by convertgls2bib are described below.

--texenc (encoding)

The character encoding of the .tex file. If omitted, the operating system’s default encoding is assumed (or the JVM’s).

--bibenc (encoding)

The character encoding of the .bib file. If omitted, the same encoding as the .tex file is assumed.

--space-sub (replacement) (or -s (replacement))

The .bib format doesn’t allow spaces in labels. If your original definitions in your .tex file have spaces, use this option to replace spaces in labels. Each space will be substituted with (replacement). The cross-referencing fields, see, seealso and alias, will also be adjusted, but any references using \gls etc will have to be substituted manually (or use a global search and replace in your text editor). If you want to strip the spaces, use an empty string for (replacement). You’ll need to delimit this according to your operating system. For example:

convertgls2bib --space-sub '' entries.tex entries.bib

372
7.1 Command Line Arguments

--ignore-sort
Omit the sort field. This is the default since bib2gls can work out a more intuitive sort value than either makeindex or xindy.

--no-ignore-sort
Don’t ignore the sort field.

--ignore-type
Omit the type field in the .bib file. You may find it more flexible not to be locked into a specific glossary type if you have a large database of entries.

--no-ignore-type
Don’t omit the type field (default unless --split-on-type).

--split-on-type (or -t)
Split the entries into separate files according to the type field. Any entries that have the type field set to \glsdefaulttype or that don’t have the type field set and there’s no default provided by the command used to define the entry (see below) then the @\langle entry\rangle data will be written to the main \langle bib file\rangle. Otherwise entries will be written to the split file (in the same directory as \langle bib file\rangle).

The split file name depends on whether or not the --split-on-category switch has also been used. With both and if the category and field values are different then the file name is \langle type\rangle-\langle category\rangle.bib otherwise it’s \langle type\rangle.bib.

Commands that have a default type are as follows:

- \newabbreviation, \newacronym, \oldacronym, \newdualentry: the default type is assumed to be abbreviations (regardless of the definition of \acronymtype or \glsxtrabbrvtype);
- \glsxtrnewsymbol: the default type is assumed to be symbols;
- \glsxtrnewnumber: the default type is assumed to be numbers;
- \newterm: the default type is assumed to be index.

This option automatically implements --ignore-type and --no-overwrite.

--no-split-on-type
Don’t split the entries into separate files according to their type (default).
7.1 Command Line Arguments

--ignore-category
Omit the category field in the .bib file.

--no-ignore-category
Don’t omit the category field (default unless --split-on-category).

--split-on-category (or -c)
Split the entries into separate files according to the category. If the category field isn’t present and there’s no default provided by the command used to define the entry (see below) then the @⟨entry⟩ data will be written to the main ⟨bib file⟩. Otherwise entries will be written to the split file (in the same directory as ⟨bib file⟩).

The split file name depends on whether or not the --split-on-type switch has also been used. With both and if the category and field values are different then the file name is ⟨type⟩⟩⟨category⟩.bib otherwise it’s ⟨category⟩.bib.

Commands that have a default category are as follows:

• \newabbreviation, \newacronym, \oldacronym, \newdualentry: the default category is assumed to be abbreviation;
• \glsxtrnewsymbol: the default category is assumed to be symbol;
• \glsxtrnewnumber: the default category is assumed to be number;
• \newterm: the default category is assumed to be index.

For example, if you have both --split-on-type and --split-on-category, then the default file name for \newabbreviation will be abbreviations-abbreviation.bib but the default file name for \newterm will be index.bib. Whereas if you only have --split-on-category and not --split-on-type, then then default file name for \newabbreviation will be abbreviation.bib.

This option automatically implements --ignore-category and --no-overwrite.

--no-split-on-category
Don’t split the entries into separate files according to their category (default).

--ignore-fields ⟨list⟩ (or -f ⟨list⟩)
Omit all the fields listed in ⟨list⟩ from the .bib file. If sort, type or category are included in the list, this will automatically implement the corresponding --ignore-sort, --ignore-type or --ignore-category option.

This option is not cumulative. If the list is empty it will unset any previous list but won’t unset any --ignore-sort, --ignore-type or --ignore-category option.

For example:
7.1 Command Line Arguments

convertgls2bib --ignore-fields 'user1,sort' entries.bib

is equivalent to:

convertgls2bib --ignore-fields user1 --ignore-sort entries.bib

and

convertgls2bib --ignore-fields 'user1,sort' --ignore-fields '' entries.bib

is equivalent to:

convertgls2bib --ignore-sort entries.bib

--preamble-only (or -p)
Stop parsing if the start of the document environment is found.

--no-preamble-only
Parse the entire file (default). Be prepared for a lot of unknown command warnings if you make convertgls2bib parse an entire document.

--absorb-see
Absorb any cross-referencing information identified with \glssee or \glsxtrindexseealso commands into the corresponding entry (default).

--no-absorb-see
Don’t absorb any cross-referencing information identified with \glssee or \glsxtrindexseealso commands.

--index-conversion (or -i)

Use @index instead of @entry if the description is empty or simply \nopostdesc or \glsxtrnopostpunc. (Only applies to terms that would otherwise be converted to @entry, such as those defined with \newglossaryentry.)

--no-index-conversion

Don’t convert @entry to @index (default).

--locale ⟨language tag⟩

Identifies the language resource file to use for convertgls2bib’s messages.
7.2 Recognised Commands

--overwrite
Allow existing .bib files to be overwitten. (Default unless --split-on-type.)

--no-overwrite
Don’t allow existing .bib files to be overwitten. (Default if --split-on-type.)

--silent
Suppress all messages except for errors.

--verbose
Display messages and warnings (default).

--debug
Display debugging messages (stack traces and other information in addition to --verbose).

--help (or -h)
Display help message and quit.

--version (or -v)
Display version information and quit.

7.2 Recognised Commands

This application recognises the commands listed below as well as some standard commands such as \newcommand. Avoid any overly complicated code within the .tex file. The \TeX parser library isn’t a \TeX engine! The .tex file doesn’t need to be a complete document, but if you want certain commands recognised from packages that the \TeX parser library supports, you’ll need to include \usepackage in the .tex file. If you want to quit parsing the .tex file at the start of the document, use the --preamble-only switch.

In all cases below, if \langle key=value list \rangle contains:

\texttt{\langle see=\{[seealsoname}\langle label(s)\}\rangle}

or

\texttt{\langle see=\{[alsoname}\langle label(s)\}\rangle}

this will be substituted with:
7.2 Recognised Commands

\seealso{\langle label(s)\rangle}

For example:
\newterm[see={\seealsoname goose}]{duck}
will be written as:
@index{duck,
  seealso = \{goose\}
}

Note that it won’t convert see={\see also\langle labels\rangle}. If you have used explicit text instead of \seealsoname or \alsoname then consider performing a global search and replace on your file using your text editor.

Additionally, if \langle key=value list \rangle contains:
\glstype={\glsdefaulttype}
then this field will be ignored. (This type value is recommended in \langle key=value list \rangle when loading files with \loadglsentries[\langle type\rangle][\langle file\rangle] to allow the optional argument to set the type. With bib2gls you can use the type option instead.)

\glsexpandfields
The base glossaries package provides:
\glsexpandfields
If present, this instructs convertgls2bib to expand all fields except for those explicitly identified by \glssetnoexpandfield. Remember that there are many commands that aren’t recognised by convertgls2bib so it may not be possible to correctly expand field values. Conversely, there are some commands that will be expanded by convertgls2bib that aren’t expandable in \TeX (such as \MakeUppercase and \char).

\glsnoexpandfields
The base glossaries package provides:
\glsnoexpandfields
If present, this instructs convertgls2bib to not expand fields unless explicitly identified by \glssetexpandfield.

\glssetexpandfield
The base glossaries package provides:
\glssetexpandfield{\langle field\rangle}
If present, this instructs convertgls2bib to expand the given field, even if \glsnoexpandfields has been used.
\texttt{\textbackslash glssetnoexpandfield}

The base \texttt{glossaries} package provides:

\begin{verbatim}
\glssetnoexpandfield{⟨field⟩}
\end{verbatim}

If present, this instructs \texttt{convertgls2bib} to not expand the given field, even if \texttt{\glsexpandfields} has been used. Unlike the default behaviour with the \texttt{glossaries} package, there are no fields switched explicitly switched off by default with \texttt{convertgls2bib}.

\texttt{\textbackslash newglossaryentry}

The base \texttt{glossaries} package provides:

\begin{verbatim}
\newglossaryentry{⟨label⟩}{⟨key=value list⟩}
\end{verbatim}

This is converted to:

\begin{verbatim}
@entry{⟨label⟩,  
  ⟨key=value list⟩}
\end{verbatim}

\texttt{\textbackslash newglossaryentry} is recognised as a synonym of \texttt{\newglossaryentry}.

\texttt{\textbackslash provideglossaryentry}

The base \texttt{glossaries} package provides:

\begin{verbatim}
\provideglossaryentry{⟨label⟩}{⟨key=value list⟩}
\end{verbatim}

This is converted to:

\begin{verbatim}
@entry{⟨label⟩,  
  ⟨key=value list⟩}
\end{verbatim}

but only if \texttt{⟨label⟩} hasn’t already been defined.

\texttt{\textbackslash longnewglossaryentry}

The base \texttt{glossaries} package provides:

\begin{verbatim}
\longnewglossaryentry{⟨label⟩}{⟨key=value list⟩}{⟨description⟩}
\end{verbatim}

This is converted to:

\begin{verbatim}
@entry{⟨label⟩,  
  ⟨key=value list⟩,  
  description = {⟨description⟩}}
\end{verbatim}
7.2 Recognised Commands

The starred version provided by the glossaries-extra package is also recognised. The un-starred version strips trailing spaces from \textit{description}. (This doesn’t add \texttt{nopostdesc}, but glossaries-extra defaults to \texttt{nopostdot}.)

\texttt{\textbackslash longprovideglossaryentry}

The base glossaries package provides:

\begin{verbatim}
\longprovideglossaryentry{⟨label⟩}{⟨key=value list⟩}{⟨description⟩}
\end{verbatim}

As above, but only if \textit{⟨label⟩} hasn’t already been defined.

\texttt{\textbackslash newterm}

The base glossaries package provides:

\begin{verbatim}
\newterm[⟨key=value list⟩]{⟨label⟩}
\end{verbatim}

(when the \texttt{index} option is used). This is converted to:

\begin{verbatim}
@index{⟨label⟩},
 ⟨key=value list⟩
\end{verbatim}

if the optional argument is present, otherwise it’s just converted to:

\begin{verbatim}
@index{⟨label⟩}
\end{verbatim}

If \texttt{--space-sub} is used and \textit{⟨label⟩} contains one or more spaces, then \texttt{name} will be set if not included in \textit{⟨key=value list⟩}. For example, if \texttt{entries.bib} contains:

\begin{verbatim}
\newterm{sea lion}
\newterm[seealso={sea lion}]{seal}
\end{verbatim}

then:

\begin{verbatim}
convertgls2bib --space-sub '-' entries.bib entries.tex
\end{verbatim}

will write the terms to \texttt{entries.tex} as:

\begin{verbatim}
@index{sea-lion,
    name = {sea lion}}
\end{verbatim}

\begin{verbatim}
@index{seal,
    seealso = {sea-lion}}
\end{verbatim}

whereas just:
convertgls2bib entries.bib entries.tex

will write the terms to entries.tex as:

\@index{sea lion}
\@index{seal,
 seealso = {sea lion} }

which will cause a problem when the .bib file is parsed by bib2gls (and will probably also cause a problem for bibliographic management systems).

\texttt{\newabbreviation}

The glossaries-extra package provides:

\texttt{\newabbreviation[(key=value list)]{(label)}{(short)}{(long)}}

This is converted to:

\texttt{\@abbreviation{(label),
  short = {(short)},
  long = {(long)},
  (key=value list) }}

if the optional argument is present, otherwise it’s converted to:

\texttt{\@abbreviation{(label),
  short = {(short)},
  long = {(long)}  }}

\texttt{\newacronym}

The base glossaries package provides:

\texttt{\newacronym[(key=value list)]{(label)}{(short)}{(long)}}

(which is redefined by glossaries-extra to use \newabbreviation).

As above but uses \texttt{@acronym} instead. The base package also provides \texttt{\oldacronym}, which emulates the way abbreviations were defined with the precursor glossary package. This has different syntax to \newacronym but is also recognised by convertgls2bib and is converted to \texttt{@acronym}. 

380
7.2 Recognised Commands

\glsxtrnewsymbol
The glossaries-extra package provides:

\glsxtrnewsymbol[\langle key=\text{value list}\rangle\{\langle label\rangle\}\{\langle symbol\rangle\}]
(when the \texttt{symbols} option is used). This is converted to:

@symbol\{(\langle label\rangle),
\texttt{name} = \{\langle symbol\rangle\}\}
if the optional argument is missing, otherwise it’s converted to:

@symbol\{(\langle label\rangle),
\texttt{name} = \{\langle symbol\rangle\},
\langle key=\text{value list}\rangle\}
unless \langle key=\text{value list}\rangle contains the \texttt{name} field, in which case it’s converted to:

@symbol\{(\langle label\rangle),
\langle key=\text{value list}\rangle\}
\texttt{newsym} is recognised as a synonym for \glsxtrnewsymbol.

\glsxtrnewnumber
The glossaries-extra package provides:

\glsxtrnewnumber[\langle key=\text{value list}\rangle\{\langle label\rangle\}]
(when the \texttt{numbers} option is used). This is converted to:

@number\{(\langle label\rangle),
\texttt{name} = \{\langle label\rangle\}\}
if the optional argument is missing, otherwise it’s converted to:

@number\{(\langle label\rangle),
\texttt{name} = \{\langle label\rangle\},
\langle key=\text{value list}\rangle\}
if \texttt{name} isn’t listed in \langle key=\text{value list}\rangle, otherwise it’s converted to:

@number\{(\langle label\rangle),
\langle key=\text{value list}\rangle\}
\texttt{newnum} is recognised as a synonym for \glsxtrnewnumber.
7.2 Recognised Commands

\newdualentry

This command isn’t provided by either glossaries or glossaries-extra but is used as an example in the glossaries user manual [14] and in the sample file sample-dual.tex that accompanies the glossaries package. Since this command seems to be used quite a bit (given the number of times it crops up on sites like \TeX{} on StackExchange), convertgls2bib also supports it unless this command is defined using \newcommand or \renewcommand in the input file. In which case the default definition will be overridden.

If the command definition isn’t overridden, then it’s converted to:

@dualabbreviationentry{%label, %short = {short}, %long = {long}, %description = {description}, (%key=value list)%}

if (%key=value list) is supplied, otherwise it’s converted to:

@dualabbreviationentry{%label, %short = {short}, %long = {long}, %description = {description} }

For example, if the original .tex file contains:

\newcommand*{\newdualentry}[5][%  \newglossaryentry{main-#2}{name={#4},%    text={#3\glsadd{#2}},%    description={#5},%    #1%  }%  \newacronym{#2}{#3\glsadd{main-#2}}{#4}%
\newdualentry{svm}% label % abbreviation % support vector machine long form % Statistical pattern recognition technique description

then the .bib file will contain:
7.2 Recognised Commands

@entry{main-svm,
    name = {support vector machine},
    description = {Statistical pattern recognition technique},
    text = {SVM\glsadd{svm}}
}

@acronym{svm,
    short = {SVM\glsadd{main-svm}},
    long = {support vector machine}
}

since \newdualentry was defined with \newcommand. However, if the original file uses \providecommand or omits the definition of \newdualentry, then the .bib file will contain:

@dualabbreviationentry{svm,
    short = {SVM},
    description = {Statistical pattern recognition technique},
    long = {support vector machine}
}
8 Examples

The example files described here can be found in the examples sub-directory. The .bib files are listed first and then sample files that use the .bib data. Make sure you have the latest versions of glossaries, mfirstuc, glossaries-extra and bib2gls if you want to try these out. (The sample-media.tex file requires at least datatool v2.28.) If you get any undefined control sequence or undefined style errors then you need to update your \TeX{} distribution. Use the --group switch when invoking bib2gls for all these examples if you want the glossaries divided into groups. The set of system calls for the document build in the examples below may require an extra \TeX{} run to ensure the PDF bookmarks are up-to-date when hyperref is used.

These files are just examples of how to use bib2gls. There are other ways of defining similar entries and sometimes alternatives are suggested. Use the code here as a starting point if you need data like this and adapt it to a format appropriate for your requirements. There are also some example documents in the Dickimaw Books gallery.

no-interpret-preamble.bib

The no-interpret-preamble.bib file contains command definitions used in some of the name fields. Although these commands aren’t used explicitly in the document, they need to be defined when the names are displayed in the document (typically in the glossary). These commands are much like the \texttt{\sortop} command described on 259 and need to be hidden from bib2gls’s interpreter. This file doesn’t contain any entry definitions and must be loaded first with \texttt{interpret-preamble=\{false\}}. The interpret-preamble.bib or interpret-preamble2.bib file can then be loaded to provide alternative definitions for bib2gls’s interpreter.

The first command is:

\texttt{\sortname{\{first name(s)\}}{\{surname\}}}

This is used in the name fields for entries containing information about a person. The aim here is for bib2gls to sort according to \texttt{\{surname\}}, \texttt{\{first name(s)\}} but for the glossary to display \texttt{\{first name(s)\} \{surname\}}. For names with a “von” part, there’s another command:

\texttt{\sortvonname{\{first name(s)\}}{\{von\}}{\{surname\}}}

which has a similar purpose. The third command is:

\texttt{\sortart{\{article\}}{\{text\}}}

384
This is the same as \sortname but is designed for titles, phrases or sentences that start with an article (such as “a” or “the”). Although it has the same definition as \sortname in this file, in the interpreted files the article part is omitted to completely ignore them in the sorting. The fourth command is:

\sortmediacreator{⟨first name(s)⟩}{⟨surname⟩}

which again is functionally the same as \sortname.

The names could be specified using \texttt{BibTeX}'s syntax instead with \texttt{bibtex-contributor-fields} to convert it, but the aim here is to show a variety of ways to use \texttt{bib2gls}. For an example of \texttt{bibtex-contributor-fields}, see the way the \texttt{cast} field in \texttt{films.bib} is dealt with.

Although the file only contains ASCII characters, it starts with an encoding line to prevent \texttt{bib2gls} from searching the entire file for it. (That's not so much of an issue with a short file, but may cause an unnecessary delay for much longer files.)

The contents of \texttt{no-interpret-preamble.bib} are as follows:

\begin{verbatim}
\% Encoding : UTF-8
@preamble{"\providecommand{\sortname}[2]{#1 #2}
\providecommand{\sortvonname}[3]{#1 #2 #3}
\providecommand{\sortart}[2]{#1 #2}
\providecommand{\sortmediacreator}[2]{#1 #2}"}
\end{verbatim}

\section*{interpret-preamble.bib}

This provides definitions of \sortname, \sortvonname, \sortart and \sortmediacreator in \texttt{@preamble} that can be picked up by the interpreter and used during sorting. Note that in this case \sortart is defined to ignore the article to completely ignore it from sorting. If you happen to have “a ⟨something⟩” and “the ⟨something⟩” where the ⟨something⟩s are identical, you may want to append the article to disambiguate them.

The contents of interpret-preamble.bib are as follows:

\begin{verbatim}
\% Encoding: UTF-8
@preamble{"\providecommand{\sortname}[2]{#2, #1}
\providecommand{\sortvonname}[3]{#2 #3, #1}
\providecommand{\sortart}[2]{#2}
\providecommand{\sortmediacreator}[2]{#2, #1}"}
\end{verbatim}

\section*{interpret-preamble2.bib}

An alternative to interpret-preamble.bib with a different definition of \sortmediacreator. This uses \texttt{\renewcommand} instead of \texttt{\providecommand} so \texttt{write-preamble=\{false\}} is required to prevent \texttt{\LaTeX} from picking up the definitions.
The contents of interpret-preamble2.bib are as follows:

```%
% Encoding: UTF-8

@preamble{"\providecommand{\sortname}[2]{#2, #1}
\providecommand{\sortvonname}[3]{#2 #3, #1}
\providecommand{\sortart}[2]{#2}
\renewcommand{\sortmediacreator}[2]{{\MakeLowercase{#2}}}"
```

**constants.bib**

The *constants.bib* file contains mathematical constants. These all use a custom entry type `@constant`, which must be aliased otherwise the entries will all be ignored. The entries all have custom fields, which also need to be aliased. For example:

```
entry-type-aliases={constant=entry},
field-aliases={
    constantname=name,
    constantsymbol=symbol,
    definition=description,
    identifier=category,
    value=user1
}
```

This setting means that, for example,

```@constant{root2,
    constantname={Pythagoras' constant},
    constantsymbol={\ensuremath{\surd2}},
    definition={the square root of 2},
    value={1.41421},
    identifier={constant}
}
```

is treated as though it was defined as:

```@entry{root2,
    name={Pythagoras' constant},
    symbol={\ensuremath{\surd2}},
    description={the square root of 2},
    user1={1.41421},
    category={constant}
}
```

This use of custom fields and entry types allows more flexibility. For example, I may have another document that uses the same `.bib` file but requires a different definition:
8 Examples: constants.bib

@number{root2,
  description={Pythagoras' constant},
  name={\ensuremath{\sqrt{2}}}
}

which can be obtained with:

entry-type-declarations={constant=number},
field-aliases={
  constantname=description,
  constantsymbol=name
}

Since the other custom fields haven’t be aliased, they’re ignored.

The custom fields are: identifier (set to constant for all the entries), constantname (the constant’s name), definition (a definition of the constant), value (the approximate numeric value of the constant), constantsymbol (the symbolic representation of the constant) and alternative (alternative symbol). There are three entries that don’t have the custom value field: zero and one (the exact value is in the constantsymbol field in both cases) and imaginary (where there’s no real number value).

I’ve provided some commands in the @preamble for constants that are represented by Latin and Greek letters. These can be defined in the document before the resource set if different notation is required. The upright Greek commands require the upgreek package.

If it’s likely that there may be a need to sort according to definition, then it would be better to use \sortart describe above:

@constant{root2,
  constantname={Pythagoras' constant},
  constantsymbol={\ensuremath{\sqrt{2}}},
  definition={\sortart{the}{square root of 2}},
  value={1.41421},
  identifier={constant}
}

Remember that this would need no-interpret-preamble.bib to ensure the command is recognised in the document.

The contents of constants.bib are as follows:

% Encoding: UTF-8
% Requires upgreek .sty

@preamble{"\providecommand{\mathrm{i}}{\mathrm{i}}
\providecommand{\mathrm{j}}{\mathbf{j}}
\providecommand{\mathrm{e}}{\mathbf{e}}
\providecommand{\uppi}{\uppi}
\providecommand{\upgamma}}
### Examples: constants.bib

```latex
\providecommand{\constantphi}{\upphi}
\providecommand{\constantlambda}{\uplambda}

@constant{pi,
  constantname={pi},
  constantsymbol={\ensuremath{\constantpi}},
  definition={the ratio of the length of the circumference of a circle to its diameter},
  value={3.14159},
  identifier={constant}
}

@constant{eulercons,
  constantname={Euler's constant},
  constantsymbol={\ensuremath{\constantgamma}},
  definition={the limit of \[
\sum_{r=1}^n\frac{1}{r}-\ln n\]
  as \(n \to \infty\)},
  value={0.57721},
  identifier={constant}
}

@constant{e,
  constantname={Euler's number},
  constantsymbol={\ensuremath{\constante}},
  definition={base of natural logarithms},
  value={2.71828},
  identifier={constant}
}

@constant{root2,
  constantname={Pythagoras' constant},
  constantsymbol={\ensuremath{\surd2}},
  definition={the square root of 2},
  value={1.41421},
  identifier={constant}
}

@constant{goldenratio,
  constantname={golden ratio},
  constantsymbol={\ensuremath{\constantphi}},
  definition={the ratio \(\frac{1+\surd5}{2}\)},
  value={1.61803},
  identifier={constant}
}

@constant{aperysconstant,
``````
8 Examples: chemicalformula.bib

constantname={Ap\'ery's constant},
constantsymbol={\ensuremath{\zeta(3)}},
definition={a special value of the Riemann zeta function},
value={1.2020569},
identifier={constant}
}

@constant{conwaysconstant,
  constantname={Conway's constant},
  constantsymbol={\ensuremath{\constantlambda}},
  definition={the invariant growth rate of all derived strings},
  value={1.30357},
  identifier={constant}
}

@constant{zero,
  constantname={zero},
  constantsymbol={\ensuremath{0}},
  definition={nothing or nil},
  identifier={constant}
}

@constant{one,
  constantname={one},
  constantsymbol={\ensuremath{1}},
  definition={single entity, unity},
  identifier={constant}
}

@constant{imaginary,
  constantname={imaginary unit},
  constantsymbol={\ensuremath{\constanti}},
  definition={defined as $\constanti^2 = -1$},
  identifier={constant},
  alternative={\ensuremath{\constantj}}
}

chemicalformula.bib

The chemicalformula.bib file contains chemical formulae. Each entry has a field that uses \ce provided by mhchem so the document will need to load that package. Since all resource files must be loaded in the preamble, it’s possible to ensure that the package is loaded using:

@preamble{"\usepackage{mhchem}"

However, it’s best just to load it in the document otherwise it won’t be available before the
The .glstex file has been loaded. Also, glossaries (and therefore glossaries-extra) must be loaded after hyperref, which usually needs to be loaded last so most packages should be loaded before glossaries-extra. Instead, I've just put a comment in the .bib file as a reminder.

All entries are defined using a custom entry type @chemical. This must be aliased using entry-type-aliases or the entries will be ignored. For example, to make @chemical behave like @symbol:

entry-type-aliases={chemical=symbol}

Remember that with the @symbol type, if the sort field is omitted bib2gls will fallback on the label by default. It can be changed to fallback on the name field instead using symbol-sort-fallback={name}. This will require the use of the interpreter if the name contains a command but bib2gls recognises the mhchem package and has a limited ability to interpret \ce. If @chemical is changed to @entry instead then the fallback for the sort will be the entry's name.

All entries only contain custom fields, which will all be ignored by bib2gls unless defined or aliased: identifier, which is set to chemical for all entries, formula, which is set to the chemical formula, and chemicalname, which is set to the chemical name. This allows the flexibility of determining whether the name or symbol field should contain the chemical formula on a per-resource basis. For example:

field-aliases={formula=name,chemicalname=description}

or

field-aliases={chemicalname=name,formula=symbol}

The contents of chemicalformula.bib are as follows:

```latex
% Encoding: UTF-8

% requires mhchem.sty

@chemical{H2O,
  formula={\ce{H2O}},
  chemicalname={water},
  identifier={chemical}
}

@chemical{Al2SO43,
  formula={\ce{Al2(SO4)3}},
  chemicalname={aluminium sulfate},
  identifier={chemical}
}

@chemical{CH3CH2OH,
  formula={\ce{CH3CH2OH}},
  chemicalname={ethanol},
}
8 Examples: chemicalformula.bib

identifier={chemical}
}

@chemical{C6H12O6,
  formula={\ce{C6H12O6}},
  chemicalname={glucose},
  identifier={chemical}
}

@chemical{CH2O,
  formula={\ce{CH2O}},
  chemicalname={formaldehyde},
  identifier={chemical}
}

@chemical{H3O+,
  formula={\ce{H3O+}},
  chemicalname={hydronium},
  identifier={chemical}
}

@chemical{SO42-,
  formula={\ce{SO4^{2-}}},
  chemicalname={sulfate},
  identifier={chemical}
}

@chemical{O2,
  formula={\ce{O2}},
  chemicalname={dioxygen},
  identifier={chemical}
}

@chemical{O,
  formula={\ce{O}},
  chemicalname={oxygen},
  identifier={chemical}
}

@chemical{OF2,
  formula={\ce{OF2}},
  chemicalname={oxygen difluoride},
  identifier={chemical}
}

@chemical{O2F2,
8 Examples: chemicalformula.bib

```latex
\begin{verbatim}
formula={\ce{O2F2}},
chemicalname={dioxygen difluoride},
identifier={chemical}
}

@chemical{OH-,
    formula={\ce{OH-}},
    chemicalname={hydroxide ion},
    identifier={chemical}
}

@chemical{AlF3,
    formula={\ce{AlF3}},
    chemicalname={aluminium trifluoride},
    identifier={chemical}
}

@chemical{Al2CoO4,
    formula={\ce{Al2CoO4}},
    chemicalname={cobalt blue},
    identifier={chemical}
}

@chemical{As4S4,
    formula={\ce{As4S4}},
    chemicalname={tetraarsenic tetrasulfide},
    identifier={chemical}
}

@chemical{C5H4NCOOH,
    formula={\ce{C5H4NCOOH}},
    chemicalname={niacin},
    identifier={chemical}
}

@chemical{C10H10O4,
    formula={\ce{C10H10O4}},
    chemicalname={ferulic acid},
    identifier={chemical}
}

@chemical{C8H10N4O2,
    formula={\ce{C8H10N4O2}},
    chemicalname={caffeine},
    identifier={chemical}
}
\end{verbatim}
```

392
The bacteria.bib file contains bacteria abbreviations. These all use the @abbreviation entry type with a short and long field.

The entries all have a custom field identifier set to bacteria. This will be ignored by bib2gls unless it's defined using \glsaddkey or \glsaddstoragekey or if it's aliased with field-aliases.

The contents of bacteria.bib are as follows:
Examples: bacteria.bib

@abbreviation{cbotulinum,  
  short={C.~botulinum},  
  long={Clostridium botulinum},  
  identifier={bacteria}  
}

@abbreviation{pputida,  
  short={P.~putida},  
  long={Pseudomonas putida},  
  identifier={bacteria}  
}

@abbreviation{cperfringens,  
  short={C.~perfringens},  
  long={Clostridium perfringens},  
  identifier={bacteria}  
}

@abbreviation{bsubtilis,  
  short={B.~subtilis},  
  long={Bacillus subtilis},  
  identifier={bacteria}  
}

@abbreviation{ctetani,  
  short={C.~tetani},  
  long={Clostridium tetani},  
  identifier={bacteria}  
}

@abbreviation{pcomposti,  
  short={P.~composti},  
  long={Planifilum composti},  
  identifier={bacteria}  
}

@abbreviation{pfimeticola,  
  short={P.~fimeticola},  
  long={Planifilum fimeticola},  
  identifier={bacteria}  
}

@abbreviation{cburnetii,  
  short={C.~burnetii},  
  long={ }
}
The baseunits.bib file contains base SI units. The entries are all defined using the custom @unit entry type. This must be aliased with entry-type-aliases otherwise bib2gls will ignore all the entries. For example:

```
entry-type-aliases={unit=symbol}
```

will make bib2gls treat the entries as though they were defined using @symbol. (Remember that @symbol entry types use the label as the fallback field for sort.)

The entries all have custom fields unitname, unitsymbol and measurement, one of which must be aliased or copied to name if @unit is aliased to an entry type that requires it. The other custom fields may be aliased or copied to symbol and description as required. The unitsymbol fields all use \si provided by the siunitx package, so that package must be loaded in the document. This is one of the small number of packages recognised by bib2gls, so it’s possible to sort according to the symbol if required.

The entries also all have a custom field identifier set to baseunit. This will be ignored by bib2gls unless it’s defined using \glsaddkey or \glsaddstoragekey or if it’s aliased with field-aliases.

The contents of baseunits.bib are as follows:

```bibtex
% Encoding: UTF-8
%
% requires siunix.sty

@unit{ampere,
  unitname={ampere},
  unitsymbol={\si{\ampere}},
  measurement={electric current},
  identifier={baseunit}}
```
@unit{kilogram,
  unitname={kilogram},
  unitsymbol={\si{\kilogram}},
  measurement={mass},
  identifier={baseunit}
}

@unit{metre,
  unitname={metre},
  unitsymbol={\si{\metre}},
  measurement={length},
  identifier={baseunit}
}

@unit{second,
  unitname={second},
  unitsymbol={\si{\second}},
  measurement={time},
  identifier={baseunit}
}

@unit{kelvin,
  unitname={kelvin},
  unitsymbol={\si{\kelvin}},
  measurement={thermodynamic temperature},
  identifier={baseunit}
}

@unit{mole,
  unitname={mole},
  unitsymbol={\si{\mole}},
  measurement={amount of substance},
  identifier={baseunit}
}

@unit{candela,
  unitname={candela},
  unitsymbol={\si{\candela}},
  measurement={luminous intensity},
  identifier={baseunit}
}
derivedunits.bib

The derivedunits.bib file is much like baseunits.bib but contains derived units and in this case the custom entry type is @measurement, which must be aliased otherwise the entries will all be ignored. The entries all have a custom field identifier set to derivedunit. This will be ignored by bib2gls unless it’s defined using \glsaddkey or \glsaddstorage-key or if it’s aliased with field-aliases.

The contents of derivedunits.bib are as follows:

```latex
% Encoding: UTF-8

% requires siunitx.sty

@measurement{area, 
    unitname={square metre}, 
    unitname={\si{\metre\textsuperscript{2}}}, 
    measurement={area}, 
    identifier={derivedunit} }

@measurement{volume, 
    unitname={cubic metre}, 
    unitsymbol={\si{\metre\textsuperscript{3}}}, 
    measurement={volume}, 
    identifier={derivedunit} }

@measurement{velocity, 
    unitname={metre per second}, 
    unitsymbol={\si{\metre\per\second}}, 
    measurement={velocity}, 
    identifier={derivedunit} }

@measurement{acceleration, 
    unitname={metre per second squared}, 
    unitsymbol={\si{\metre\per\square\second}}, 
    measurement={acceleration}, 
    identifier={derivedunit} }

@measurement{density, 
    unitname={ampere per square metre}, 
    unitsymbol={\si{\ampere\per\square\metre}}, 
    measurement={density}, 
    identifier={derivedunit} }
```

397
@measurement{luminance,
    unitname={candela per square metre},
    unitsymbol={\si{\candela\per\square\metre}},
    measurement={luminance},
    identifier={derivedunit}
}

@measurement{specificvolume,
    unitname={cubic metre per kilogram},
    unitsymbol={\si{\cubic\metre\per\kilogram}},
    measurement={specific volume},
    identifier={derivedunit}
}

@measurement{concentration,
    unitname={mole per cubic metre},
    unitsymbol={\si{\mole\per\cubic\metre}},
    measurement={concentration},
    identifier={derivedunit}
}

@measurement{wavenumber,
    unitname={per metre},
    unitsymbol={\si{\per\metre}},
    measurement={wave number},
    identifier={derivedunit}
}

people.bib

The people.bib file contains details about people. The name fields contain custom commands provided in no-interpret-preamble.bib and interpret-preamble.bib. Remember that if no-interpret-preamble.bib is loaded first, the definitions provided in that file will be the one in use in the document. The interpret-preamble.bib file then needs to be loaded to provide the definitions for bib2gls's interpreter.

The information for each person is supplied in an @entry type. There are some non-standard fields: born, died and othername. These fields will be ignored unless keys are provided (using \glsaddkey or \glsaddstoragekey) or the fields are aliased (using field-aliases). The born and died fields have dates that are almost in the default en-GB locale format with the JRE locale provider, but they include a tilde ~ to prevent awkward line breaks. By default bib2gls's interpreter converts ~ to the non-breaking space character 0xA0 which isn't recognised by the date format. This can easily be fixed with the --break-space switch which will interpret ~ as a normal breakable space (0x20), so with that switch sort={date}
or sort={date-reverse} can be used on either of those fields. However, the CLDR has a slightly different default format than the JRE for dates with en-GB, so it’s probably simplest to actually specify the required format.

An alternative approach would be to provide a command that can be modified in the document to adjust the date style. For example, the born field could be specified as:

\begin{verbatim}
nborn={\formatdate{13}{7}{100}{BC}}\end{verbatim}

The definition provided for the document could then be, for example:

\begin{verbatim}
providecommand{\formatdate}[4]{\DTMdisplaydate{#3}{#2}{#1}{-1} #4}
\end{verbatim}

(where \DTMdisplaydate is provided by the datetime2 package) and a definition could be provided for bib2gls’s interpreter, for example:

\begin{verbatim}
providecommand{\formatdate}[4]{#1/#2/#3 #4}
\end{verbatim}

This would need the date format set. For example, date-sort-format={d/M/y G}.

Some of the entries, such as caesar, have a first field. In those cases the first field is slightly different from the name field (for example, ”Gaius” is omitted in caesar’s first field). The other entries don’t have a first field. They can simply have the name copied to first with the replicate-fields option (so that the full name is shown on first use) or the first field can be ignored with ignore-fields (so all entries will use the text field on first use). The replicate-override option can be used to force the name field to be copied to the first field, even if the first field is already set. Alternatively, with replicate-override={true} and replicate-fields={first=name}, the first field be copied to the name field. For consistency, the first fields use the same custom commands as used in the name field.

There’s one name with a “von” part. In this case the name field is set to:

\begin{verbatim}
\sortvonname{Manfred}{von}{Richthofen}
\end{verbatim}

which will come under the “V” letter group since \sortvonname is defined as (von) (surname), (first name(s))

If you prefer that this name should come under “R” instead, then you need to adjust the definition of \sortvonname:

\begin{verbatim}
@preamble{"providecommand{\sortname}[2]{#2, #1}
 providecommand{\sortvonname}[3]{#3, #1 #2}"
\end{verbatim}

An alternative approach would be to format the names using BstT\LaTeX{}’s contributor syntax and use \texttt{bibtex-contributor-fields=\{name\}} to convert them.

There are also some synonyms provided with \texttt{@index} entry types that have the alias field to redirect to the main entry. These don't include a description or any of the other fields as that would be redundant. All the information can be found in the main entry.

Except for the aliases, the entries have a custom field identifier set to person. This will be ignored by bib2gls unless it’s defined using \texttt{\glsaddkey} or \texttt{\glsaddstoragekey} or if it’s aliased with field-aliases.

The contents of people.bib are as follows:
@entry{caesar,
    name={\sortname{Gaius Julius}{Caesar}},
    first={\sortname{Julius}{Caesar}},
    text={Caesar},
    description={Roman politician and general},
    born={13-\text{July} 100 BC},
    died={15-\text{March} 44 BC},
    identifier={person}
}

@entry{wellesley,
    name={\sortname{Arthur}{Wellesley}},
    text={Wellington},
    description={Anglo-Irish soldier and statesman},
    born={1-\text{May} 1769 AD},
    died={14-\text{September} 1852 AD},
    othername={1st Duke of Wellington},
    identifier={person}
}

@index{wellington,
    name={Wellington},
    alias={wellesley},
    identifier={person}
}

@entry{bonaparte,
    name={\sortname{Napoleon}{Bonaparte}},
    text={Bonaparte},
    description={French military and political leader},
    born={15-\text{July} 1769 AD},
    died={5-\text{May} 1821 AD},
    identifier={person}
}

@entry{alexander,
    name={Alexander III of Macedon},
    text={Alexander},
    description={Ancient Greek king of Macedon},
    born={20-\text{July} 356 BC},
    died={10-\text{June} 323 BC},
    othername={Alexander the Great},
    identifier={person}
}
@index{alexandergreat,
  name={Alexander the Great},
  alias={alexander},
  identifier={person}
}

@entry{vonrichthofen,
  name={\sortvonnname{Manfred}{von}{Richthofen}},
  text={von Richthofen},
  description={Prussian ace fighter pilot in the German Air Force
during World War-I},
  born={2-May 1892 AD},
  died={21-April 1918 AD},
  othername={The Red Baron},
  identifier={person}
}

@index{redbaron,
  name={\sortart{The}{Red Baron}},
  alias={vonrichthofen},
  identifier={person}
}

@entry{dickens,
  name={\sortname{Charles}{Dickens}},
  text={Dickens},
  description={English writer and social critic},
  born={7-February 1812 AD},
  died={9-June 1870 AD},
  identifier={person}
}

@entry{chandler,
  name={\sortname{Raymond}{Chandler}},
  text={Chandler},
  description={American-British novelist and screenwriter},
  born={23-July 1888 AD},
  died={26-March 1959 AD},
  identifier={person}
}

@entry{hammett,
  name={\sortname{Samuel Dashiell}{Hammett}},
  first={\sortname{Dashiell}{Hammett}},
  text={Hammett},
  identifier={person}
}
8 Examples: people.bib

description={American author, screenwriter and political activist},
born={27-May 1894 AD},
died={10-January 1961 AD},
identifier={person}
}

@entry{christie,
  name={\sortname{Dame Agatha Mary Clarissa}{Christie}},
  first={\sortname{Agatha}{Christie}},
  text={Christie},
  othername={Lady Mallowan},
  description={English crime novelist and playwright},
  born={15-September 1890 AD},
  died={12-January 1976 AD},
  identifier={person}
}

@entry{landon,
  name={\sortname{Christopher Guy}{Landon}},
  first={\sortname{Christopher}{Landon}},
  text={Landon},
  description={British novelist and screenwriter},
  born={29-March 1911 AD},
  died={26-April 1961 AD},
  identifier={person}
}

@entry{tolkien,
  name={\sortname{John Ronald Reuel}{Tolkien}},
  first={\sortname{J.R.R.}{Tolkien}},
  text={Tolkien},
  description={English writer, poet, philologist, and university professor},
  born={3-January 1892 AD},
  died={2-September 1973 AD},
  identifier={person}
}

@entry{baum,
  name={\sortname{Lyman Frank}{Baum}},
  first={\sortname{L.-Frank}{Baum}},
  text={Baum},
  description={American author},
  born={15-May 1856 AD},
  died={6-May 1919 AD},

402
8 Examples: people.bib

identifier={person}
}

@entry{mackenzie,
    name={\sortname{Compton}{Mackenzie}},
    text={Mackenzie},
    description={English-born Scottish writer, cultural
                commentator, raconteur and Scottish nationalist},
    born={17-January 1883 AD},
    died={30-November 1972 AD},
    identifier={person}
}

@entry{maclean,
    name={\sortname{Alistair}{MacLean}},
    text={MacLean},
    description={Scottish novelist},
    born={21-April 1922 AD},
    died={2-February 1987 AD},
    identifier={person}
}

@entry{dick,
    name={\sortname{Philip K.}{Dick}},
    text={Dick},
    description={American science fiction writer},
    born={16-December 1928 AD},
    died={2-March 1982 AD},
    identifier={person}
}

@entry{story,
    name={\sortname{Jack Trevor}{Story}},
    text={Story},
    description={British novelist},
    born={30-March 1917 AD},
    died={5-December 1991 AD},
    identifier={person}
}

@entry{greene,
    name={\sortname{Henry Graham}{Green}},
    first={\sortname{Graham}{Greene}},
    text={Green},
    description={English novelist},
    born={2-October 1904 AD},
    }
The books.bib file contains details about books. As above, the entries use custom commands provided in no-interpret-preamble.bib and interpret-preamble.bib or interpret-preamble2.bib. The entries all have a custom field `identifier` set to `book` and other custom fields `author` and `year`. These will be ignored by bib2gls unless they’re defined using `\glsaddkey` or `\glsaddstoragekey` or if they’re aliased with `field-aliases`.

There are other ways in which this data could be specified. For example, the `description` field could contain a brief summary (or “log line”). The `author` field could use BibTeX’s syntax instead with `bibtex-contributor-fields` to convert it. Alternatively, the entries could be defined using standard BibTeX entry types that are all aliased to `@bibtexentry`.

The contents of books.bib are as follows:

```
\% Encoding: UTF-8

@entry{ataleoftwocities,
    name={\sortart{A}{Tale of Two Cities}},
    description={novel by Charles Dickens},
    identifier={book},
    author={\sortmediacreator{Charles}{Dickens}},
    year={1859}
}

@entry{bleakhouse,
    name={Bleak House},
    description={novel by Charles Dickens},
    identifier={book},
    author={\sortmediacreator{Charles}{Dickens}},
    year={1852}
}

@entry{thebigsleep,
    name={\sortart{The}{Big Sleep}},
    description={novel by Raymond Chandler},
    identifier={book},
    author={\sortmediacreator{Raymond}{Chandler}},
    year={1939}
}

@entry{thelonggoodbye,
    name={\sortart{The}{Long Goodbye}},
```
8 Examples: books.bib

description={novel by Raymond Chandler},
identifier={book},
author={\sortmediacreator{Raymond}{Chandler}},
year={1953}
}

@entry{redharvest,
name={Red Harvest},
description={novel by Dashiell Hammett},
identifier={book},
author={\sortmediacreator{Dashiell}{Hammett}},
year={1929}
}

@entry{murderontheorientexpress,
name={Murder on the Orient Express},
description={novel by Agatha Christie},
identifier={book},
author={\sortmediacreator{Agatha}{Christie}},
year={1934}
}

@entry{whydidnttheyaskevans,
name={Why Didn't They Ask Evans?},
description={novel by Agatha Christie},
identifier={book},
author={\sortmediacreator{Agatha}{Christie}},
year={1934}
}

@entry{icecoldinalex,
name={Ice Cold in Alex},
description={novel by Christopher Landon},
identifier={book},
author={\sortmediacreator{Christopher}{Landon}},
year={1957}
}

@entry{thehobbit,
name={\sortart{The}{Hobbit}},
description={novel by J.R.R. Tolkien},
identifier={book},
author={\sortmediacreator{J.R.R.}{Tolkien}},
year={1937}
}
@entry{thelordoftherings,
  name=\{sortart{The}{Lord of the Rings}\},
  description=\{novel by J.R.R. Tolkien\},
  identifier=\{book\},
  author=\{sortmediacreator{J.R.R.}{Tolkien}\},
  year=\{1954\}
}

@entry{thewonderfulwizardofoz,
  name=\{sortart{The}{Wonderful Wizard of Oz}\},
  description=\{novel by L. Frank Baum\},
  identifier=\{book\},
  author=\{sortmediacreator{L. Frank}{Baum}\},
  year=\{1900\}
}

@entry{whiskygalore,
  name=\{Whisky Galore\},
  description=\{novel by Compton Mackenzie\},
  identifier=\{book\},
  author=\{sortmediacreator{Compton}{Mackenzie}\},
  year=\{1947\}
}

@entry{whereeaglesdare,
  name=\{Where Eagles Dare\},
  description=\{novel by Alistair MacLean\},
  identifier=\{book\},
  author=\{sortmediacreator{Alistair}{MacLean}\},
  year=\{1967\}
}

@entry{icestationzebra,
  name=\{Ice Station Zebra\},
  description=\{novel by Alistair MacLean\},
  identifier=\{book\},
  author=\{sortmediacreator{Alistair}{MacLean}\},
  year=\{1963\}
}

@entry{ubik,
  name=\{Ubik\},
  description=\{novel by Philip K. Dick\},
  identifier=\{book\},
  author=\{sortmediacreator{Philip K.}{Dick}\},
  year=\{1969\}
}
8 Examples: films.bib

}

@entry{doandroidsdreamofelectricSheep,
  name={Do Androids Dream of Electric Sheep?},
  description={novel by Philip K. Dick},
  identifier={book},
  author={\sortmediacreator{Philip K.}{Dick}},
  year={1968}
}

@entry{thetroublewithharry,
  name={\sortart{The}{Trouble with Harry}},
  description={novel by Jack Trevor Story},
  identifier={book},
  author={\sortmediacreator{Jack Trevor}{Story}},
  year={1950}
}

@entry{brightonrock,
  name={Brighton Rock},
  description={novel by Graham Greene},
  identifier={book},
  author={\sortmediacreator{Graham}{Greene}},
  year={1938}
}

films.bib

The films.bib file contains details about films. As above, the entries use custom commands provided in no-interpret-preamble.bib and interpret-preamble.bib. The entries all have a custom field identifier set to film and other custom fields cast, director and year. These will be ignored by bib2gls unless they’re defined using \glsaddkey or \glsaddstoragekey or if they’re aliased with field-aliases.

This example file references entries defined in books.bib through the use of the special ext1. prefix. To avoid a label conflict films.bib prefixes all labels with film. rather than relying on label-prefix. This ensures that both books.bib and films.bib can be loaded in the same resource set (otherwise they’d have to be loaded in separate resource sets with different prefixes). Remember that you can use \glsxtrnewgls. For example:

\glsxtrnewgls{film.}{\film}

This means you can do, for example, just \film{bladerunner} if you want to reference a film without worrying about the prefix.

As with all the example files, there are other ways in which to specify the data, depending on your requirements. For example, the director field could use \tex{}’s contributor syntax
(as the cast field does). Some of the films actually had more than one director but only one is listed per film in this sample file for simplicity. Similarly, the cast field only contains the principal actors rather than the complete list. The book on which the film is based could be contained in a cross-reference field or a custom basedon field.

The book “Do Androids Dream of Electric Sheep?” referenced at the end of the “Blade Runner” film’s description ends with a question mark. (Similarly for “Why Didn’t They Ask Evans?”) If the description field is simply set as:

```
description={a film starring Harrison Ford, Rutger Hauer
and Sean Young loosely based on the novel
\gls{ext1.doandroidsdreamofelectricsheep}},
```

then the postdot package option will produce an odd result as the inserted full stop immediately follows the question mark. This is an awkward situation. One possibility is to explicitly put the full stop at the end of the description field for all the other entries and omit it for the problematic entries, but this interferes with the possibility of a category-dependent post-description hook.

Another option is to put \nopostdesc in the problematic entries. For example:

```
description={a film starring Harrison Ford, Rutger Hauer
and Sean Young loosely based on the novel
\gls{ext1.doandroidsdreamofelectricsheep}\nopostdesc},
```

Be careful with this as it will completely suppress the post-description hook. A third possibility is to use \glsxtrnopostpunc instead:

```
description={a film starring Harrison Ford, Rutger Hauer
and Sean Young loosely based on the novel
\gls{ext1.doandroidsdreamofelectricsheep}\glsxtrnopostpunc},
```

This doesn’t interfere with the post-description hook but if a hook is provided the post-punctuation may then be required. In both of the above two cases, strip-trailing-nopost could be used to remove the suppression commands from the description fields if a hook is defined. However this doesn’t deal with hooks that only conditionally append text.

The best solution is with glossaries-extra v1.23+ which provides \glsxtrrestorepostpunc for use in the category post-description hooks that counteracts \glsxtrnopostpunc. This can be placed inside a conditional, as used in sample-media.tex, and does nothing if \glsxtrnopostpunc doesn’t occur in the description field. (Note that \glsxtrrestorepostpunc can’t be used to counteract \nopostdesc, since that completely suppresses the hook.)

The contents of films.bib are as follows:

```
% Encoding: UTF-8

@entry{film.thebigsleep,
   name={\sortart{The}{Big Sleep}},
   description={a film based on the novel
```
8 Examples: films.bib

\gls{ext1.thebigsleep},
cast={Humphrey Bogart and Lauren Bacall},
identifier={film},
year={1946},
director={\sortmediacreator{Howard}{Hawks}}
}

@entry{film.thelonggoodbye,
  name={\sortart{The}{Long Goodbye}},
  description={a film based on the novel \gls{ext1.thelonggoodbye}},
  cast={Elliott Gould and Nina van Pallandt},
  identifier={film},
  year={1973},
  director={\sortmediacreator{Robert}{Altman}}
}

@entry{film.murderontheorientexpress,
  name={Murder on the Orient Express},
  description={a film based on the novel \gls{ext1.murderontheorientexpress}},
  cast={Albert Finney and Lauren Bacall and Ingrid Bergman},
  identifier={film},
  director={\sortmediacreator{Sidney}{Lumet}},
  year={1974}
}

@entry{film.whydidnttheyaskevans,
  name={Why Didn't They Ask Evans?},
  description={a film based on the novel \gls{ext1.whydidnttheyaskevans}\glsxtrnopostpunc},
  cast={Francesca Annis and John Gielgud and Bernard Miles},
  identifier={film},
  director={\sortmediacreator{John}{Davies}},
  year={1980}
}

@entry{film.icecoldinalex,
  name={Ice Cold in Alex},
  description={a film based on the novel \gls{ext1.icecoldinalex}},
  cast={John Mills and Anthony Quayle and Sylvia Sims},
  identifier={film},
  year={1958},
  director={\sortmediacreator{J. Lee}{Thompson}}
}
@entry{film.anunexpectedjourney, 
  name={\sortart{The}{Hobbit}: \sortart{An}{Unexpected Journey}}, 
  description={a film based on the novel \gls{ext1.thehobbit}}, 
  cast={Martin Freeman and Ian McKellen and Richard Armitage}, 
  identifier={film}, 
  year={2012}, 
  director={\sortmediacreator{Peter}{Jackson}} 
}

@entry{film.desolationofsmaug, 
  name={\sortart{The}{Hobbit}: \sortart{The}{Desolation of Smaug}}, 
  description={a film based on the novel \gls{ext1.thehobbit}}, 
  cast={Ian McKellen and Martin Freeman and Richard Armitage}, 
  identifier={film}, 
  year={2013}, 
  director={\sortmediacreator{Peter}{Jackson}} 
}

@entry{film.thebattleoffivearmies, 
  name={\sortart{The}{Hobbit}: \sortart{The}{Battle of Five Armies}}, 
  description={a film based on the novel \gls{ext1.thehobbit}}, 
  cast={Ian McKellen and Martin Freeman and Richard Armitage}, 
  identifier={film}, 
  year={2014}, 
  director={\sortmediacreator{Peter}{Jackson}} 
}

@entry{film.thefellowshipofthering, 
  name={\sortart{The}{Lord of the Rings}: \sortart{The}{Fellowship of the Ring}}, 
  description={a film based on the novel \gls{ext1.thelordoftherings}}, 
  cast={Elijah Wood and Ian McKellen and Orlando Bloom}, 
  identifier={film}, 
  year={2001}, 
  director={\sortmediacreator{Peter}{Jackson}} 
}

@entry{film.thetwotowers, 
  name={\sortart{The}{Lord of the Rings}: 

410
8 Examples: films.bib

\sortart{The}{Two Towers},
description={a film based on the novel
\gls{ext1.thelordoftherings}},
cast={Elijah Wood and Ian McKellen and Viggo Mortensen},
identifier={film},
year={2002},
director={\sortmediacreator{Peter}{Jackson}}
}

@entry{film.thereturnoftheking,
name={\sortart{The}{Lord of the Rings}}:
\sortart{The}{Return of the King},
description={a film based on the novel
\gls{ext1.thelordoftherings}},
cast={Elijah Wood and Viggo Mortensen and Ian McKellen},
identifier={film},
year={2003},
director={\sortmediacreator{Peter}{Jackson}}
}

@entry{film.thewizardofoz,
name={\sortart{The}{Wizard of Oz}},
description={a film based on the novel
\gls{ext1.thewonderfulwizardofoz}},
cast={Judy Garland},
identifier={film},
year={1939},
director={\sortmediacreator{Victor}{Fleming}}
}

@entry{film.whiskygalore,
name={Whisky Galore!},
description={a film based on the novel
\gls{ext1.whiskygalore}},
cast={Basil Radford and Joan Greenwood},
identifier={film},
year={1949},
director={\sortmediacreator{Alexander}{Mackendrick}}
}

@entry{film.whereeaglesdare,
name={Where Eagles Dare},
description={a film based on the novel
\gls{ext1.whereeaglesdare}},
cast={Richard Burton and Clint Eastwood and Mary Ure},
identifier={film},
year={1968},
director={\sortmediacreator{J. Lee Thompson}{}}
}
8 Examples: films.bib

year={1968},
director={\sortmediacreator{Brian G.}{Hutton}}
}

@entry{film.icestationzebra,
name={Ice Station Zebra},
description={a film based on the novel \gls{ext1.icestationzebra}},
cast={Rock Hudson and Ernest Borgnine},
identifier={film},
year={1968},
director={\sortmediacreator{John}{Sturges}}
}

@entry{film.bladerunner,
name={Blade Runner},
description={a film loosely based on the novel \gls{ext1.doandroidsdreamofelectricsheep}\glsxtrnopostpunc},
cast={Harrison Ford and Rutger Hauer and Sean Young},
identifier={film},
year={1982},
director={\sortmediacreator{Ridley}{Scott}}
}

@entry{film.thetroublewithharry,
name={\sortart{The}{Trouble with Harry}},
description={a film based on the novel \gls{ext1.thetroublewithharry}},
cast={John Forsythe and Shirley MacLaine},
identifier={film},
year={1955},
director={\sortmediacreator{Alfred}{Hitchcock}}
}

@entry{film.brightonrock,
name={Brighton Rock},
description={a film based on the novel \gls{ext1.brightonrock}},
cast={Richard Attenborough and Hermione Baddeley
and William Hartnell},
identifier={film},
year={1947},
director={\sortmediacreator{John}{Boutling}}
}
citations.bib

The citations.bib file is actually a \texttt{BibTeX} file, but it can be parsed by \texttt{bib2gls} if the \texttt{BibTeX} entry types are converted to \texttt{@bibtexentry}, which can easily be done with:

\begin{verbatim}
entry-type-aliases={\GlsXtrBibTeXEntryAliases}
\end{verbatim}

The field names will also need to be defined or aliased. For example:

\begin{verbatim}
field-aliases={title=name}
\end{verbatim}

If \texttt{bib2gls} is then run with \texttt{--cite-as-record} any \texttt{\cite} commands found in the \texttt{.aux} file will be treated as ignored records. The \texttt{@preamble} provides a formatting command that's used by both \texttt{BibTeX} and \texttt{bib2gls}, so \texttt{\providecommand} is required rather than \texttt{\newcommand} as it will appear in both the \texttt{.bbl} and the \texttt{.glstex} files. (In general it's best to use \texttt{\providecommand} rather than \texttt{\newcommand} in the \texttt{@preamble} but in this case it's essential.)

The contents of citations.bib are as follows:

\begin{verbatim}
\% Encoding: UTF-8
\@preamble{"\providecommand{\titlefmt}[1]{`#1'}"}
\@article{duck2018,
  author = {Dickie Duck and Jos'\'{e} Arara and Polly Parrot},
  title = {Avian friendship},
  journal = {Fowl Times},
  year = 2018,
  volume = 7,
  number = 5,
  pages = "1032--5"
}
\@book{duck2016,
  author = {Dickie Duck},
  title = {Feathered stunt doubles: \titlefmt{The Birds} and other films},
  publisher = {Duck Duck Goose},
  year = 2016
}
\@book{macaw,
  author = {Prof Macaw},
  title = {Annotated notes on the \titlefmt{Duck and Goose} chronicles},
  publisher = {Duck Duck Goose},
  year = 2012
}
\end{verbatim}
8 Examples: mathgreek.bib

@book{ing,
    author = {Bor Ing},
    title = {	exttt{Duck and Goose}: an allegory for modern times?},
    publisher = {Duck Duck Goose},
    year = 2010
}

@article{parrot,
    author = {Polly Parrot and Dickie Duck},
    title = {	exttt{Duck and Goose} Cheat Sheet for Students},
    journal = {Fowl Times},
    year = 2013,
    volume = 2,
    number = 10,
    pages = "15--23"
}

@book{parrot2012,
    author = {A Parrot},
    title = {My Friend is a Duck},
    publisher = {Duck Duck Goose},
    year = 2012
}

@book{quackalot,
    author = {Sir Quackalot},
    title = {The Adventures of Duck and Goose},
    publisher = {Duck Duck Goose},
    year = 2011
}

mathgreek.bib

The mathgreek.bib file contains Greek letters for use in maths mode. These are all defined with \texttt{@symbol}, which means that by default the \texttt{sort} field will be obtained from the label not from the \texttt{name} field. However, if you want to sort by the \texttt{name} field (for example, with \texttt{sort-field=\{name\}}) the \LaTeX{} parser library recognises all the mathematical Greek letter commands provided in the \LaTeX{} kernel. Additionally it recognises \texttt{\omicron} which isn’t provided by \LaTeX{} (the symbol can be reproduced with a lower case Latin “\texttt{o}”). Note that \texttt{glossaries-extra-bib2gls} (\texttt{glossaries-extra v1.27+}) provides all the missing Greek letters (such as \texttt{\omicron}).

The .bib file could just use \texttt{o}:

\texttt{@symbol{omicron,}
8 Examples: mathgreek.bib

```latex
\begin{verbatim}
name={\ensuremath{o}},
description={omicron},
identifier={mathgreek}
\end{verbatim}
```

but this means that if bib2gls sorts according to the name field using a letter sort, this entry will come before all the other Greek letters since the character “o” has Unicode value 0x6F whereas, for example, mathematical italic small alpha (α) has Unicode value 0x1D6FC. This means that for sorting purposes it’s better to use \omicron:

```latex
@symbol{omicron,
   name={\ensuremath{\omicron}},
   description={omicron},
   identifier={mathgreek}
}
```

but \LaTeX needs a definition for this, so it’s provided in the @preamble:

```latex
@preamble{"\providecommand{\omicron}{o}"

(With glossaries-extra v1.27+, this is no longer needed.) The \LaTeX parser library and glossaries-extra–bib2gls similarly provide the missing upper case Greek letters, and these can be dealt with in the same way.

The contents of mathgreek.bib are as follows:

```%
\begin{verbatim}
\% Encoding: UTF-8

@preamble{"\providecommand{\omicron}{o}"

@symbol{alpha,
   name={\ensuremath{\alpha}},
   description={alpha},
   identifier={mathgreek}
}

@symbol{beta,
   name={\ensuremath{\beta}},
   description={beta},
   identifier={mathgreek}
}

@symbol{gamma,
   name={\ensuremath{\gamma}},
   description={gamma},
   identifier={mathgreek}
}

@symbol{delta,
   name={\ensuremath{\delta}},
   description={delta},
   identifier={mathgreek}
}
```
```
name={\ensuremath{\delta}},
  description={\delta},
  identifier={mathgreek}
}

@symbol{\varepsilon,
  name={\ensuremath{\varepsilon}},
  description={\epsilon (variant)},
  identifier={mathgreek}
}

@symbol{\zeta,
  name={\ensuremath{\zeta}},
  description={\zeta},
  identifier={mathgreek}
}

@symbol{\eta,
  name={\ensuremath{\eta}},
  description={\eta},
  identifier={mathgreek}
}

@symbol{\theta,
  name={\ensuremath{\theta}},
  description={\theta},
  identifier={mathgreek}
}

@symbol{\iota,
  name={\ensuremath{\iota}},
  description={\iota},
  identifier={mathgreek}
}

@symbol{\kappa,
  name={\ensuremath{\kappa}},
  description={\kappa},
  identifier={mathgreek}
}

@symbol{\lambda,
  name={\ensuremath{\lambda}},
  description={\lambda},
  identifier={mathgreek}
}
@symbol{mu,
  name={\texttt{\textbackslash ensuremath{\textmu}}},
  description={\textmu},
  identifier={mathgreek}}

@symbol{nu,
  name={\texttt{\textbackslash ensuremath{\textnu}}},
  description={\textnu},
  identifier={mathgreek}}

@symbol{xi,
  name={\texttt{\textbackslash ensuremath{\textxi}}},
  description={\textxi},
  identifier={mathgreek}}

@symbol{omicron,
  name={\texttt{\textbackslash ensuremath{\textomicron}}},
  description={\textomicron},
  identifier={mathgreek}}

@symbol{pi,
  name={\texttt{\textbackslash ensuremath{\textpi}}},
  description={\textpi},
  identifier={mathgreek}}

@symbol{rho,
  name={\texttt{\textbackslash ensuremath{\textrho}}},
  description={\textrho},
  identifier={mathgreek}}

@symbol{varsigma,
  name={\texttt{\textbackslash ensuremath{\textvarsigma}}},
  description={\textvarsigma (variant)},
  identifier={mathgreek}}

@symbol{sigma,
  name={\texttt{\textbackslash ensuremath{\textsigma}}},
  description={\textsigma},
  identifier={mathgreek}}
8 Examples: mathgreek.bib

description={sigma},
identifier={mathgreek}
}

@symbol{tau,
  name={\ensuremath{\tau}},
  description={\tau},
  identifier={mathgreek}
}

@symbol{upsilon,
  name={\ensuremath{\upsilon}},
  description={\upsilon},
  identifier={mathgreek}
}

@symbol{varphi,
  name={\ensuremath{\varphi}},
  description={\phi (variant)},
  identifier={mathgreek}
}

@symbol{chi,
  name={\ensuremath{\chi}},
  description={\chi},
  identifier={mathgreek}
}

@symbol{psi,
  name={\ensuremath{\psi}},
  description={\psi},
  identifier={mathgreek}
}

@symbol{omega,
  name={\ensuremath{\omega}},
  description={\omega},
  identifier={mathgreek}
}

@symbol{epsilon,
  name={\ensuremath{\epsilon}},
  description={\epsilon},
  identifier={mathgreek}
}
@symbol{vartheta,
    name={\ensuremath{\vartheta}},
    description={\(\theta\) (variant)},
    identifier={mathgreek}
}

@symbol{varkappa,
    name={\ensuremath{\varkappa}},
    description={\(\kappa\) (variant)},
    identifier={mathgreek}
}

@symbol{phi,
    name={\ensuremath{\phi}},
    description={\(\phi\)},
    identifier={mathgreek}
}

@symbol{varrho,
    name={\ensuremath{\varrho}},
    description={\(\rho\) (variant)},
    identifier={mathgreek}
}

@symbol{varpi,
    name={\ensuremath{\varpi}},
    description={\(\pi\) (variant)},
    identifier={mathgreek}
}

**bigmathsymbols.bib**

The bigmathsymbols.bib file contains mathematical symbols that have a large version in display mode. As with mathgreek.bib the entries are defined using `@symbol`. This example file requires the stix package as not all of the commands are provided by the \LaTeX kernel. This file also has a preamble:

@preamble{"\providecommand{\bigoperatornamefmt}[1]{%\$\displaystyle#1\textstyle#1$}
\providecommand{\nary}[1]{\textstyle\raisebox{-0.5ex}{\textstyle#1}}"}

The first command \texttt{\bigoperatornamefmt{(text)}} is used in the name field to display both the in-line and display versions of the symbol. The \LaTeX parser library only has a limited ability to interpret this as not all the symbols have Unicode in-line and large versions. In some cases, such as the integral symbol \(\int\), there is only a small version. (A large version
would require construction from \(0x2320, 0x23AE\) and \(0x2321\), which is too complicated in this context.) However, the interpreter works well enough to guess at the widest name if `set-widest` is used. There’s no advantage in sorting according to the `name` field here, unless a custom rule is provided, as the Unicode symbols are scattered about different blocks. Better approaches are to sort according to document use (`sort={use}`) or to sort according to the `description` field.

The other custom command is \texttt{\nary{⟨text⟩}} to provide semantic markup for “\(n\)-ary”. This could be defined without an argument:

\begin{verbatim}
\providecommand{\nary}{⟨n$\$-ary⟩}
\end{verbatim}

but providing an argument will allow \texttt{\nary{n}} to work with first letter upper-casing in the event that the `description` field has a case-change applied (otherwise it would end up as “\(N\)-ARY”). Of course, it may be that no case-change should be applied, but this example is just for illustrative purposes.

As with the other sample `.bib` files, each entry is given a custom `identifier` field, which by default will be ignored. In this case, `identifier` is either set to `naryoperator` (for \(n\)-ary operators) or `integral` for integrals.

The contents of `bigmathsymbols.bib` are as follows:

```latex
\begin{verbatim}
\% Encoding: UTF-8
\%
\% requires stix.sty
\%\preamble{"\providecommand{\bigoperatornamefmt}[1]{%$\displaystyle#1\textstyle#1$}\providecommand{\nary}[1]{#1$-ary"}}
\symbol{bigsqcap},
\quad name={\bigoperatornamefmt{\bigsqcap}},
\quad text={\bigsqcap},
\quad description={\nary{n} square intersection operator},
\quad identifier={naryoperator}
\}
\symbol{bigsqcup},
\quad name={\bigoperatornamefmt{\bigsqcup}},
\quad text={\bigsqcup},
\quad description={\nary{n} square union operator},
\quad identifier={naryoperator}
\}
\symbol{sum},
\quad name={\bigoperatornamefmt{\sum}},
\quad text={\sum},
\quad description={\nary{n} summation},
\quad identifier={naryoperator}
\end{verbatim}
```

420
8 Examples: bigmathsymbols.bib

@symbol{prod,
    name={\bigoperatornamefmt{\prod}},
    text={\prod},
    description={\nary{n} product},
    identifier={naryoperator}
}

@symbol{coprod,
    name={\bigoperatornamefmt{\coprod}},
    text={\coprod},
    description={\nary{n} coproduct},
    identifier={naryoperator}
}

@symbol{bigcap,
    name={\bigoperatornamefmt{\bigcap}},
    text={\bigcap},
    description={\nary{n} intersection},
    identifier={naryoperator}
}

@symbol{bigcup,
    name={\bigoperatornamefmt{\bigcup}},
    text={\bigcup},
    description={\nary{n} union},
    identifier={naryoperator}
}

@symbol{bigodot,
    name={\bigoperatornamefmt{\bigodot}},
    text={\bigodot},
    description={\nary{n} circled dot operator},
    identifier={naryoperator}
}

@symbol{bigoplus,
    name={\bigoperatornamefmt{\bigoplus}},
    text={\bigoplus},
    description={\nary{n} circled plus operator},
    identifier={naryoperator}
}

@symbol{bigotimes,
    name={\bigoperatornamefmt{\bigotimes}},
}
8 Examples: bigmathsymbols.bib

text={\bigotimes},
description={\text{nary\{n\} circled times operator}},
identifier={naryoperator}
}

@symbol{biguplus,
    name={\bigoperatornamefmt{\biguplus}},
    text={\biguplus},
    description={\text{nary\{n\} union operator with plus}},
    identifier={naryoperator}
}

@symbol{bigvee,
    name={\bigoperatornamefmt{\bigvee}},
    text={\bigvee},
    description={\text{nary\{n\} logical or}},
    identifier={naryoperator}
}

@symbol{bigwedge,
    name={\bigoperatornamefmt{\bigwedge}},
    text={\bigwedge},
    description={\text{nary\{n\} logical and}},
    identifier={naryoperator}
}

@symbol{int,
    name={\bigoperatornamefmt{\int}},
    text={\int},
    description={integral},
    identifier={integral}
}

@symbol{iint,
    name={\bigoperatornamefmt{iint}},
    text={\iint},
    description={double integral},
    identifier={integral}
}

@symbol{iiint,
    name={\bigoperatornamefmt{\iiint}},
    text={\iiint},
    description={triple integral},
    identifier={integral}
}
mathsrelations.bib

The mathsrelations.bib file contains mathematical relational symbols. These use the maths shift character $ in the name field and just the symbol in the text field. This just illustrates an alternative way of defining symbols. Since \ensuremath isn’t used, commands like \gls must be explicitly placed in maths mode. For example, $\gls{leq}$ rather than simply \gls{leq}. The custom identifier field is set to relation.

The contents of mathsrelations.bib are as follows:

% Encoding: UTF-8

@symbol{leq,
  name={$\leq$},
  text={$\leq$},
  description={less than or equal to},
  identifier={relation}
}

@symbol{less,
  name={$<$},
  text={<},
  description={less than},
  identifier={relation}
}
Examples: mathsrelations.bib

@symbol{ll,  
  name={$\ll$},  
  text={$\ll$},  
  description={much less than},  
  identifier={relation}}

@symbol{geq,  
  name={$\geq$},  
  text={$\geq$},  
  description={greater than or equal to},  
  identifier={relation}}

@symbol{greater,  
  name={$>$},  
  text={>},  
  description={greater than},  
  identifier={relation}}

@symbol{gg,  
  name={$\gg$},  
  text={\gg},  
  description={much greater than},  
  identifier={relation}}

@symbol{equals,  
  name={$=$},  
  text={=},  
  description={equals},  
  identifier={relation}}

@symbol{neq,  
  name={$\neq$},  
  text={\neq},  
  description={not equals},  
  identifier={relation}}

@symbol{approx,  
  name={$\approx$},  
  text={\approx}}
The `binaryoperators.bib` file contains mathematical binary operators. The format is much like the above `mathsrelations.bib` file. The custom `identifier` field is set to `binaryoperator`. The contents of `binaryoperators.bib` are as follows:

```plaintext
@symbol{plus,
  name={$+$},
  text={+},
  description={addition},
  identifier={binaryoperator}
}

@symbol{minus,
  name={$-$},
  text={-},
  description={subtraction},
  identifier={binaryoperator}
}

@symbol{times,
  name={$\times$},
  text={\times},
  description={multiplication},
  identifier={binaryoperator}
}
```
8 Examples: unaryoperators.bib

unaryoperators.bib

The unaryoperators.bib file contains mathematical unary operators. As above, this again uses `@symbol` to define the symbols, but in this case `\ensuremath` is used in the `name` field and there’s no `text` field. I’ve also used `\mathord` to ensure the symbol is treated as a unary (rather than binary) operator, except for the `\forall` entry which is already defined as an ordinary maths symbol.

The contents of unaryoperators.bib are as follows:

```bibtex
% Encoding: UTF-8

@symbol{div,
  name={$\div$},
  text={$\div$},
  description={division},
  identifier={binaryoperator}
}

@symbol{factorial,
  name={\ensuremath{\mathord{!}}},
  description={factorial},
  identifier={unary}
}

@symbol{unaryplus,
  name={\ensuremath{\mathord{+}}},
  description={plus},
  identifier={unary}
}

@symbol{unaryminus,
  name={\ensuremath{\mathord{-}}},
  description={minus},
  identifier={unary}
}

@symbol{forall,
  name={\ensuremath{\forall}},
  description={for all},
  identifier={unary}
}
```
mathsobjects.bib

The mathsobjects.bib file contains entries related to mathematical objects (sets, spaces, vectors and matrices). This provides some custom formatting commands in the preamble:

\setfmt{⟨symbol⟩}

which is used to format ⟨symbol⟩ as a set,

\setcontentsfmt{⟨contents⟩}

which is used to format the set contents,

\setmembershipfmt{⟨variable(s)⟩}{⟨condition⟩}

which is used to format the set membership criteria,

\setcardfmt{⟨maths⟩}

which is used to format the cardinality of a set. (Note this uses \texttt{\textbackslash vert} not | as in some of the earlier examples.)

\nums capefmt{⟨symbol⟩}

which is used to format ⟨symbol⟩ as a number space,

\transposefmt{⟨maths⟩}

which is used to format matrix and vector transposes,

\invfmt{⟨maths⟩}

which is used to format inverses,

\vecfmt{⟨symbol⟩}

which is used to format ⟨symbol⟩ as a vector, and

\mtxfmt{⟨symbol⟩}

which is used to format ⟨symbol⟩ as a matrix. These commands are intended for use with \glsxtrfmt, but \setmembershipfmt causes a problem as it has two arguments and \glsxtrfmt requires the control sequence to have exactly one argument. This means employing a little trick. A command with just one argument is provided:

\setmembershiponeargfmt{⟨variable(s)⟩}{⟨condition⟩}

that requires the actual two arguments to be supplied inside #1. The outer grouping is removed and the two-argument \setmembershipfmt command is applied:

\providecommand{\setmembershiponeargfmt}[1]{\setmembershipfmt#1}
This means that the entry needs to be referenced in the document using:

\glsxtrfmt{setmembership}{{\langle\text{variable(s)}\rangle}\{\langle\text{condition}\rangle\}}

The simplest thing to do here is to provide a wrapper command in the document, for example:

\newcommand*{\setmembership}{\glsxtrfmt{setmembership}{{#1}\{#2}}}  

Now this can be used as:

\setmembership{{\langle\text{variable(s)}\rangle}\{\langle\text{condition}\rangle\}}

There are essentially two types of entry defined in this file: entries that demonstrate the formatting for the objects and entries that represent specific objects. In the first case there’s a custom \texttt{format} field that’s set to the control sequence name of the relevant semantic command. If this field is defined or aliased then it can be used with \glsxtrfmt (as in the example above).

In both cases there’s a custom \texttt{identifier} field that reflects the type of object: \texttt{set} for sets, \texttt{numspace} for number spaces, \texttt{matrix} for matrices or vectors.

Be careful with the set cardinality example. Remember that nested links cause problems and the glossaries-extra manual advises against using commands like \texttt{gls} or \glsxtrfmt within link text and that includes within the \langle\text{text}\rangle argument of \glsxtrfmt. See sample-maths.tex for suggested usage.

Some of the \texttt{description} fields use \texttt{\sortart}, so no-interpret-preamble.bib and interpret-preamble.bib are also needed.

The contents of mathsobjects.bib are as follows:

% Encoding: UTF-8

% requires amssymb.sty

@preamble{
"providecommand{\setfmt}[1]{\mathcal{#1}}
providecommand{\setcontentsfmt}[1]{\{#1\}}
providecommand{\setmembershipfmt}[2]{\setcontentsfmt{#1: #2}}
providecommand{\setmembershiponeargfmt}[1]{\setmembershipfmt#1}
providecommand{\setcardfmt}[1]{\lvert#1\rvert}
providecommand{\numspacefmt}[1]{\mathbb{#1}}
providecommand{\transposefmt}[1]{#1^T}
providecommand{\invfmt}[1]{#1^{-1}}
providecommand{\vecfmt}[1]{\boldsymbol{#1}}
providecommand{\mtxfmt}[1]{\boldsymbol{#1}}"

@symbol{set,
  name={\ensuremath{\setfmt{S}}},
  description={\sortart{a}{set}},
  format={\setfmt},
  identifier={set}}
8 Examples: mathsobjects.bib

@symbol{setcontents,  
  name={\ensuremath{\setcontentsfmt{\ldots}}},  
  description={set contents},  
  format={setcontentsfmt},  
  identifier={set}  
}

@symbol{setmembership,  
  name={\ensuremath{\setmembershipfmt{\vecfmt{x}}{\ldots}}},  
  description={set membership},  
  format={setmembershiponeargfmt},  
  identifier={set}  
}

@symbol{setcard,  
  name={\ensuremath{\setcardfmt{\setfmt{S}}}},  
  description={\sortart{the}{cardinality of $\setfmt{S}$}},  
  format={setcardfmt},  
  identifier={set}  
}

@symbol{numberspace,  
  name={\ensuremath{\numspacefmt{S}}},  
  description={\sortart{a}{number space}},  
  format={numspacefmt},  
  identifier={numberspace}  
}

@symbol{naturalnumbers,  
  name={\ensuremath{\numspacefmt{N}}},  
  description={\sortart{the}{set of natural numbers}},  
  identifier={numberspace}  
}

@symbol{integernumbers,  
  name={\ensuremath{\numspacefmt{Z}}},  
  description={\sortart{the}{set of integers}},  
  identifier={numberspace}  
}

@symbol{rationalnumbers,  
  name={\ensuremath{\numspacefmt{Q}}},  
  description={\sortart{the}{set of rational numbers}},  
  identifier={numberspace}  
}
8 Examples: mathsobjects.bib

@symbol{algebraicnumbers,
    name={\ensuremath{\numspacefmt{A}}},
    description={\sortart{the}{set of algebraic numbers}},
    identifier={numberspace}
}

@symbol{realnumbers,
    name={\ensuremath{\numspacefmt{R}}},
    description={\sortart{the}{set of real numbers}},
    identifier={numberspace}
}

@symbol{imaginarynumbers,
    name={\ensuremath{\numspacefmt{I}}},
    description={\sortart{the}{set of imaginary numbers}},
    identifier={numberspace}
}

@symbol{complexnumbers,
    name={\ensuremath{\numspacefmt{C}}},
    description={\sortart{the}{set of complex numbers}},
    identifier={numberspace}
}

@symbol{emptyset,
    name={\ensuremath{\emptyset}},
    description={\sortart{the}{empty set}},
    identifier={set}
}

@symbol{universalset,
    name={\ensuremath{\setfmt{U}}},
    description={\sortart{the}{universal set}},
    identifier={set}
}

@symbol{transpose,
    name={\ensuremath{\transposefmt{\vecfmt{x}}}},
    description={\sortart{the}{transpose of $\vecfmt{x}$}},
    format={transposefmt},
    identifier={matrix}
}

@symbol{inverse,
    name={\ensuremath{\invfmt{\mtxfmt{M}}}},
}
8 Examples: miscsymbols.bib

description={\sortart{the}inverse of $\mtxfmt{M}$},
format={invfmt},
identifier={matrix}
}

@symbol{vector,
    name={\ensuremath{\vecfmt{v}}},
    description={\sortart{a}vector},
    format={vecfmt},
    identifier={matrix}
}

@symbol{matrix,
    name={\ensuremath{\mtxfmt{M}}},
    description={\sortart{a}matrix},
    format={mtxfmt},
    identifier={matrix}
}

@symbol{0vec,
    name={\ensuremath{\vecfmt{0}}},
    description={\sortart{the}vector of 0s},
    identifier={matrix}
}

@symbol{1vec,
    name={\ensuremath{\vecfmt{1}}},
    description={\sortart{the}vector of 1s},
    identifier={matrix}
}

@symbol{identitymatrix,
    name={\ensuremath{\mtxfmt{I}}},
    description={\sortart{the}identity matrix},
    identifier={matrix}
}

miscsymbols.bib

The miscsymbols.bib file contains text symbols provided by the marvosym and ifsym packages. The ifsym package needs to be loaded with the weather option to provide the weather commands. Unfortunately both packages define \Sun and \Lightning, which causes a conflict. See sample-textsymbols.tex for a workaround. Alternatively, you can load ifsym without the weather option and use the internal definition of ifsym’s \Sun and \Lightning commands:
@icon{sun,
   icon={\textweathersymbol{16}},
   description={sunny},
   identifier={weather}
}

@icon{lightning,
   icon={\textweathersymbol{26}},
   description={thunderstorm},
   identifier={weather}
}

This removes the conflict, and \Sun and \Lightning are as defined by marvosym.

This file uses a custom entry type @icon, which must be aliased to a recognised entry identifier otherwise the entries will all be ignored. For example:

entry-type-aliases={icon=symbol}

There are three types of symbols defined: media controls, information and weather. They have the custom identifier field set to mediacontrol, information and weather, respectively. There are two other custom fields: icon and icondescription. These will need to be aliased to name and description.

Neither of these packages are recognised by bib2gls, which means that set-widest won’t be able to determine the widest name nor is this data suitable for sorting according to the icon field (or its alias). Instead, either sort by label (which is the default for @symbol) or by the description. If you want to use one of the alttree styles you can still use set-widest, but it will have to use the fallback command. Alternatively, you can omit set-widest and explicitly use \glsFindWidestTopLevelName.

The contents of micsymbols.bib are as follows:

% Encoding : UTF-8
% requires marvosym.sty and ifsym.sty

@icon{forward,
   icon={\Forward},
   icondescription={play},
   identifier={mediacontrol}
}

@icon{forwardtoindex,
   icon={\ForwardToIndex},
   icondescription={next track},
   identifier={mediacontrol}
}

@icon{rewindtoindex,
8 Examples: miscsymbols.bib

icon={\RewindToIndex},
icondescription={back to start of track},
identifier={mediacontrol}
}

@icon{rewind,
icon={\Rewind},
icondescription={rewind},
identifier={mediacontrol}
}

@icon{bicycle,
icon={\Bicycle},
icondescription={bicycle route},
identifier={information}
}

@icon{coffeecup,
icon={\Coffeecup},
icondescription={caf\'e},
identifier={information}
}

@icon{info,
icon={\Info},
icondescription={information centre},
identifier={information}
}

@icon{gentsroom,
icon={\Gentsroom},
icondescription={Gents},
identifier={information}
}

@icon{ladiesroom,
icon={\Ladiesroom},
icondescription={Ladies},
identifier={information}
}

@icon{wheelchair,
icon={\Wheelchair},
icondescription={wheelchair access provided},
identifier={information}
}
@icon{football,
  icon={\Football},
  icondescription={football stadium},
  identifier={information}
}

@icon{recycling,
  icon={\Recycling},
  icondescription={recycling centre},
  identifier={information}
}

@icon{cloud,
  icon={\Cloud},
  icondescription={cloudy},
  identifier={weather}
}

@icon{fog,
  icon={\Fog},
  icondescription={foggy},
  identifier={weather}
}

@icon{thinfog,
  icon={\ThinFog},
  icondescription={misty},
  identifier={weather}
}

@icon{hail,
  icon={\Hail},
  icondescription={hail},
  identifier={weather}
}

@icon{sun,
  icon={\Sun},
  icondescription={sunny},
  identifier={weather}
}

@icon{lightning,
  icon={\Lightning},
  icondescription={thunderstorm},
  identifier={information}
}
The markuplanguages.bib file includes a mixture of @entry and @abbreviation definitions. A custom command is provided in @preamble to tag the letters in the long field that are used to form the abbreviation. This simply does its argument and is provided in case it's not set up in the document. If you do want to enable tagging using \GlsXtrEnableInitialTagging, remember that this command must be used before the abbreviations are defined, which means before the resource file is input with \GlsXtrLoadResources. Similarly, the abbreviation style must be set before the abbreviations are defined.

For convenience @string is also used to define a .bib variable, which may be appended to fields using the .bib concatenation character #. As with the other sample .bib files, there's a custom field identifier which will be ignored unless defined or aliased.

The empty braces at the start some of the fields are there to protect against first letter uppercasing within \TeX, where it might cause a problem. (For example, with the glossname attribute.)

The contents of markuplanguages.bib are as follows:

% Encoding: UTF-8
8 Examples: markuplanguages.bib

@preamble{\providecommand{\abbrvtag}[1]{#1}}
@string{markuplang="\abbrvtag{m}arkup \abbrvtag{l}anguage"}

@entry{TeX,  
   name={{}\TeX},  
   description={a format for describing complex type and page layout  
                often used for mathematics, technical, and academic publications},  
   identifier={markuplanguage}  
}

@entry{LaTeX,  
   name={{{}\LaTeX}},  
   description={a format of \gls{TeX} designed to separate  
                content from style},  
   identifier={markuplanguage}  
}

@entry{markdown,  
   name={markdown},  
   description={a lightweight markup language with plain text  
                formatting syntax},  
   identifier={markuplanguage}  
}

@abbreviation{xml,  
   short={XML},  
   long={e\abbrvtag{x}tensible }#markuplang,  
   description={a markup language that defines a set of rules for  
               encoding documents},  
   identifier={markuplanguage}  
}

@abbreviation{html,  
   short={HTML},  
   long={\abbrvtag{h}yper\abbrvtag{t}ext }#markuplang,  
   description={the standard markup language for creating web pages},  
   identifier={markuplanguage}  
}

@abbreviation{mathml,  
   short={MathML},  
   long={\abbrvtag{m\NoCaseChange{ath}}}#markuplang,  
   description={markup language for describing mathematical notation},  
   identifier={markuplanguage}  
}
The `usergroups.bib` file requires either \TeX\LaTeX\ or \LaTeX\ as some of the entry labels use non-ASCII characters. This file has a mixture of \texttt{@abbreviation} and \texttt{@index} entries. It also uses \texttt{@string} for convenience and provides a custom command \texttt{\dash} in \texttt{@preamble}. Each entry is the name of a \TeX\ user group: the international \TeX\ Users Group (\texttt{TUG}) and all the local groups. Most of them have an abbreviated name, so they’re defined with \texttt{@abbreviation}. There are a few without an abbreviation, so they’re defined with \texttt{@index} instead. There’s one alias. (The information was obtained from \texttt{TUG}’s user groups page \cite{18}.)

As with the other examples, there are some custom fields which will be ignored if they aren’t defined or aliased: \texttt{identifier} (set to \texttt{texusergroup}), \texttt{language} (a comma-separated list of language tags) and \texttt{translation} (provides a translation if the user group name isn’t in English).

Not all entries have a \texttt{translation} field. If it’s omitted, then the user group name is in English, otherwise it’s in the first language listed in the \texttt{language} field. Most of the language tags are just the ISO 639-1 language code, but a few of them include the ISO 3166-1 region code as well.

The contents of `usergroups.bib` are as follows:

```bibtex
% Encoding: UTF-8
% Requires XeLaTeX/LuaLaTeX for non-ASCII labels
@string{tug={\TeX\ Users Group}}
@preamble{"\providecommand{\dash}{{,\,-\,-,}}"}
@abbreviation{TUG,
    short={TUG},
...
8 Examples: usergroups.bib

long=tug,
language={en},
identifier={texusergroup}
}

@abbreviation{bgTeX,
  short={bgTeX},
  long={Bulgarian \LaTeX\ Users Group},
  language={bg},
  identifier={texusergroup}
}

@abbreviation{latex-br,
  short={latex-br},
  long={Grupo de Usuários},
  language={pt-BR},
  identifier={texusergroup},
  translation={Brazilian}#tug
}

@abbreviation{CTeX,
  short={CTeX},
  long={Chinese \TeX\ Society},
  identifier={texusergroup},
  language={zh}
}

@abbreviation{CSTUG,
  short={CSTUG},
  long={Československé sdružení uživatelů TeXu, z.-s.},
  language={cs},
  identifier={texusergroup},
  translation={Czech Republic}#tug
}

@abbreviation{DANTE,
  short={DANTE e.V.},
  long={Deutschsprachige Anwendervereinigung \TeX\ e.V.},
  language={de},
  identifier={texusergroup},
  translation={German Speaking}#tug
}

@abbreviation{DKTUG,
  short={DK-TUG},
  long={Danish}#tug,
language={da},
  identifier={texusergroup}
}

@index{EUG,
  name={Estonian User Group},
  language={et},
  identifier={texusergroup}
}

@abbreviation{CervanTeX,
  short={CervanTeX},
  long={Grupo de Usuarios de \TeX\ Hispanohablantes},
  language={es},
  identifier={texusergroup},
  translation={Spanish Speaking }#tug
}

@abbreviation{TirantloTeX,
  short={Tirant lo \TeX},
  long={Catalan }#tug,
  language={ca},
  identifier={texusergroup}
}

@abbreviation{GUTenberg,
  short={GUTenberg},
  long={Groupe francophone des utilisateurs de \TeX},
  language={fr},
  identifier={texusergroup},
  translation={French Speaking }#tug
}

@abbreviation{UKTUG,
  short={UK-TUG},
  long={UK }#tug,
  language={en-GB},
  identifier={texusergroup}
}

@abbreviation{εϕτ,
  short={εϕτ},
  long={Σύλλογος Ελλήνων Φίλων του \TeX},
  language={el},
  identifier={texusergroup},
  translation={Greek \TeX\ Friends}
@abbreviation{MaTeX,  
  short={MaTeX},  
  long={Magyar \TeX\ Egyesület},  
  language={hu},  
  identifier={texusergroup},  
  translation={Hungarian }#tug
}

@abbreviation{ITALIC,  
  short={ITALIC},  
  long={Irish \TeX\ and \LaTeX\ In-print Community},  
  language={en-IE,en-GB},  
  identifier={texusergroup}
}

@abbreviation{ÍsTeX,  
  short={ÍsTeX},  
  long={Vefur íslenskra \TeX\ notenda},  
  language={is},  
  identifier={texusergroup},  
  translation={Icelandic }#tug
}

@abbreviation{GuIT,  
  short={GuIT},  
  long={Gruppo Utilizzatori Italiani di \TeX},  
  language={it},  
  identifier={texusergroup},  
  translation={Italian }#tug
}

@abbreviation{KTS,  
  short={KTS},  
  identifier={texusergroup},  
  long={Korean \TeX\ Society},  
  language={ko}
}

@index{KTUG,  
  alias={KTS},  
  identifier={texusergroup}
}

@index{LTVG,
8 Examples: usergroups.bib

name={Lietuvos \TeX'o Vartotojų Grupė},
language={lt},
identifier={texusergroup},
translation={Lithuanian }#tug
}

@index{mxTeX,
name={\TeX\ México},
language={es-MX},
identifier={texusergroup},
translation={Mexican }#tug
}

@abbreviation{NTG,
short={NTG},
long={Nederlandstalige \TeX\ Gebruikersgroep},
language={nl},
identifier={texusergroup},
translation={Netherlands }#tug
}

@index{NTUG,
name={Nordic \TeX\ Users Group},
language={da,et,fi,fo,is,nb,nn,sv},
identifier={texusergroup}
}

@abbreviation{GUST,
short={GUST},
long={Polska Grupa Użytkowników Systemu \TeX},
language={pl},
identifier={texusergroup},
translation={Polish }#tug
}

@abbreviation{GUTpt,
short={GUTpt},
long={Grupo de Utilizadores de \TeX},
language={pt},
identifier={texusergroup},
translation={Portuguese }#tug
}

@abbreviation{VietTUG,
short={VietTUG},
long={Vietnamese }#tug,
animals.bib

The animals.bib file contains entries defined using @entry. As with the above example .bib files, there's a custom identifier field that will be ignored unless defined or aliased.

The contents of animals.bib are as follows:

```bibtex
@entry{duck,
  name={duck},
  description={a waterbird with webbed feet},
  identifier={animal}
}

@entry{parrot,
  name={parrot},
  description={mainly tropical bird with bright plumage},
  identifier={animal}
}

@entry{goose,
  name={goose},
  plural={geese},
  description={a large waterbird with a long neck, short legs, webbed feet and a short broad bill},
  identifier={animal}
}

@entry{swan,
  name={swan},
  description={a large waterbird with a long flexible neck, short legs, webbed feet and a broad bill},
  identifier={animal}
}
```
@entry{chicken,  
    name={chicken},  
    description={a domestic fowl},  
    identifier={animal}  
}  

@entry{aardvark,  
    name={aardvark},  
    description={nocturnal African burrowing mammal},  
    identifier={animal}  
}  

@entry{zebra,  
    name={zebra},  
    description={wild African horse with black-and-white stripes},  
    identifier={animal}  
}  

@entry{armadillo,  
    name={armadillo},  
    description={nocturnal insectivore with large claws},  
    identifier={animal}  
}  

@entry{zander,  
    name={zander},  
    description={large freshwater perch},  
    identifier={animal}  
}  

@entry{hedgehog,  
    name={hedgehog},  
    description={small nocturnal mammal with a spiny coat and short legs},  
    identifier={animal}  
}  

@entry{seal,  
    name={seal},  
    description={sea-dwelling fish-eating mammal with flippers},  
    identifier={animal}  
}  

@entry{sealion,  
    name={sea lion},  
    description={a large type of \gls{seal}},  
}
8 Examples: minerals.bib

minerals.bib

The minerals.bib file contains entries defined using @entry. As with the above example .bib files, there's a custom identifier field that will be ignored unless defined or aliased. The contents of minerals.bib are as follows:

% Encoding: UTF-8

@entry{quartz,
  name={quartz},
  description={hard mineral consisting of silica},
  identifier={mineral}
}

@entry{corundum,
  name={corundum},
  description={crystalline form of aluminium oxide},
  identifier={mineral}
}

@entry{beryl,
  name={beryl},
  description={composed of beryllium aluminium cyclosilicate},
  identifier={mineral}
}

@entry{amethyst,
  name={amethyst},
  description={purple variety of \textit{\acs{quartz}}},
  identifier={mineral}
}

@entry{chalcedony,
  name={chalcedony},
  description={cryptocrystalline variety of \textit{\acs{quartz}}},
  identifier={mineral}
}

@entry{citrine,
  name={citrine},
  description={yellow variety of \textit{\acs{quartz}}},
  identifier={mineral}
}
@entry{aquamarine,  
  name={aquamarine},  
  description={light blue variety of \gls{beryl}},  
  identifier={mineral}  
}

@entry{aragonite,  
  name={aragonite},  
  description={a crystal form of calcium carbonate},  
  identifier={mineral}  
}

@entry{calcite,  
  name={calcite},  
  description={a crystal form of calcium carbonate},  
  identifier={mineral}  
}

@entry{vaterite,  
  name={vaterite},  
  description={a crystal form of calcium carbonate},  
  identifier={mineral}  
}

@entry{bakerite,  
  name={bakerite},  
  description={a borosilicate mineral},  
  identifier={mineral}  
}

@entry{bilinite,  
  name={bilinite},  
  description={an iron sulfate mineral},  
  identifier={mineral}  
}

@entry{biotite,  
  name={biotite},  
  description={a common phyllosilicate mineral},  
  identifier={mineral}  
}

@entry{cobaltite,  
  name={cobaltite},  
  description={a sulfide mineral composed of cobalt, arsenic and
sulfur},
   identifier={mineral}
}

@entry{cyanotrichite,
   name={cyanotrichite},
   description={a hydrous copper aluminium sulfate mineral},
   identifier={mineral}
}

@index{lettsomite,
   alias={cyanotrichite},
   identifier={mineral}
}

@entry{diamond,
   name={diamond},
   description={a metastable allotrope of carbon},
   identifier={mineral}
}

@entry{dolomite,
   name={dolomite},
   description={an anhydrous carbonate mineral},
   identifier={mineral}
}

@entry{quetzalcoatlite,
   name={quetzalcoatlite},
   description={a rare tellurium oxysalt mineral},
   identifier={mineral}
}

@entry{vulcanite,
   name={vulcanite},
   description={a rare copper telluride mineral},
   identifier={mineral}
}

vegetables.bib

The vegetables.bib file contains entries defined using @entry and an entry defined with @index with just the alias field. As with the above example .bib files, there’s a custom identifier field that will be ignored unless defined or aliased.

The contents of vegetables.bib are as follows:
@entry{cabbage,  
  name={cabbage},  
  description={vegetable with thick green or purple leaves},  
  identifier={vegetable}  
}

@entry{brussels-sprout,  
  name={Brussels sprout},  
  description={small leafy green vegetable buds},  
  identifier={vegetable}  
}

@entry{artichoke,  
  name={artichoke},  
  description={a variety of thistle cultivated as food},  
  identifier={vegetable}  
}

@entry{cauliflower,  
  name={cauliflower},  
  description={type of cabbage with edible white flower head},  
  identifier={vegetable}  
}

@entry{spinach,  
  name={spinach},  
  description={green, leafy vegetable},  
  identifier={vegetable}  
}

@entry{marrow,  
  name={marrow},  
  description={long white-fleshed gourd with green skin},  
  identifier={vegetable}  
}

@entry{courgette,  
  name={courgette},  
  description={immature fruit of a vegetable \gls{marrow}},  
  identifier={vegetable}  
}

@index{zucchini,  
  name={zucchini},  
  447
8 Examples: terms.bib

```bash
alias={courgette},
identifier={vegetable}
```

terms.bib

The terms.bib file contains entries defined using @index. Unlike the above sample .bib files, there are no custom fields here.

The contents of terms.bib are as follows:

```bash
% Encoding: UTF-8

@index{mineral}
@index{vegetable}
@index{animal}
@index{film}
@index{book}
@index{bacteria,
text={bacterium},
plural={bacteria}}
@index{chemical,
name={chemical formula},
plural={chemical formulae}}
@index{baseunit,
name={base SI unit}}
@index{derivedunit,
name={derived SI unit}}
@index{person,
plural={people}}
@index{markuplanguage,
name={markup language}}
@index{mediacontrol,
name={media control}}
@index{information}
@index{weather}
```

448
topics.bib

The topics.bib file contains entries defined using \texttt{\@index}. Again there are no custom fields here.

The contents of topics.bib are as follows:

\begin{verbatim}
% Encoding : UTF-8
\@index{information}
\@indexplural{mediacontrol,text={media control}}
\@indexplural{weather,text={weather symbol}}
\end{verbatim}

sample-hierarchical.tex

This example uses the terms.bib, animals.bib, minerals.bib and vegetables.bib files to create a hierarchical glossary. These are specified with the resource option:

\begin{verbatim}
src={terms,animals,minerals,vegetables}
\end{verbatim}

The custom \texttt{identifier} field is aliased to the \texttt{parent} field since it conveniently matches the labels of the animal, mineral and vegetable entries in the terms.bib file:

\begin{verbatim}
field-aliases={identifier=parent}
\end{verbatim}

The default \texttt{selection} setting means that only those terms referenced in the document and their dependencies are selected. The referenced entries simply have “1” in the location list as it’s only a trivial single-paged example.

The dependencies that haven’t actually been referenced in the document don’t have a location list. (The “seal” entry is a dependency, but it’s also been referenced in the document, so it has a location list.) The “quartz”, “beryl” and “marrow” entries are dependencies because they occur in the description of some of the referenced entries. Normally this would mean that they have no location list after the first \texttt{\LaTeX+\bib2gls+\LaTeX} build but once the glossary has been created the references to those dependent entries in the descriptions will create records and so on the next \texttt{\bib2gls+\LaTeX} they will also have location lists. This would make the complete document build:

\begin{verbatim}
pdflatex sample-hierarchical
\bib2gls --group sample-hierarchical
pdflatex sample-hierarchical
\bib2gls --group sample-hierarchical
pdflatex sample-hierarchical
\end{verbatim}

However, in this example I’ve decided to ignore any records created in the glossary.
This means that the document build is the usual \LaTeX+\bib2gls+\LaTeX. I've used the treegroup style so I need to invoke \bib2gls with the --group switch. This creates letter groups for the top-level entries. Note that sub-entries never have letter groups.

The complete code is listed below. The document build is:

```tex
\texttt{pdflatex sample-hierarchical}
\texttt{bib2gls --group sample-hierarchical}
\texttt{pdflatex sample-hierarchical}
```

The complete document is shown in figure 8.1.

```tex
\documentclass[12pt,a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage[record,% use \bib2gls
nostyles,% don't load default styles
postdot,% add a full stop after the description
% load glossary-tree.sty and patch styles:
stylesmods={tree},
style=treegroup]{glossaries-extra}
\GlsXtrLoadResources[
  src={terms,animals,minerals,vegetables},% data these .bib files
  field-aliases={identifier=parent}]
\begin{document}
Some sample terms: \gls{duck}, \gls{sealion}, \gls{armadillo},
\gls{seal}, \gls{aardvark}, \gls{amethyst}, \gls{aquamarine},
\gls{diamond}, \gls{dolomite}, \gls{chalcedony}, \gls{citrine},
\gls{quetzalcoatlite}, \gls{cabbage}, \gls{cauliflower},
\gls{artichoke}, \gls{courgette}.
\GlsXtrSetDefaultNumberFormat{glsignore}% ignore records in the glossary
\printunsrtglossary
\end{document}
```

\texttt{sample-nested.tex}

As discussed in section 1.2 there are three ways of creating logical divisions when displaying the entries through the use of the \texttt{type}, \texttt{group} and \texttt{parent} fields. In general, hierarchical glossaries are created with the \texttt{parent} field and an appropriate glossary style (as in the previous \texttt{sample-hierarchical.tex} example).
Some sample terms: duck, sea lion, armadillo, seal, aardvark, amethyst, aquamarine, diamond, dolomite, chalcedony, citrine, quetzalcoatlite, cabbage, cauliflower, artichoke, courgette.

Glossary

A
animal
aardvark nocturnal African burrowing mammal. 1
armadillo nocturnal insectivore with large claws. 1
duck a waterbird with webbed feet. 1
sea lion a large type of seal. 1
seal sea-dwelling fish-eating mammal with flippers. 1

M
mineral
amethyst purple variety of quartz. 1
aquamarine light blue variety of beryl. 1
beryl composed of beryllium aluminium cyclosilicate.
chalcedony cryptocrystalline variety of quartz. 1
citrine yellow variety of quartz. 1
diamond a metastable allotrope of carbon. 1
dolomite an anhydrous carbonate mineral. 1
quartz hard mineral consisting of silica.
quartz anhydrous carbonate mineral. 1
quetzalcoatlite a rare tellurium oxysalt mineral. 1

V
vegetable
artichoke a variety of thistle cultivated as food. 1
cabbage vegetable with thick green or purple leaves. 1
cauliflower type of cabbage with edible white flower head. 1
courgette immature fruit of a vegetable marrow. 1
marrow long white-fleshed gourd with green skin.
This example creates a hierarchical effect but the entries don’t actually have a hierarchical structure as none of them have the `parent` field set. Instead, what were the child entries in `sample-hierarchical.tex` now have the `type` field set. The hierarchical effect is achieved with \printunsrtinnerglossary (which requires at least glossaries-extra v1.44).

The \printunsrtinnerglossary command is unsuitable for use with tabular-like styles, such as long, and can be problematic with list styles. However, those styles aren’t suitable for hierarchical glossaries anyway.

Normally, hierarchy is achieve through definitions like:

```
@index{animal}
@entry{duck, name={duck},
  description={a waterbird with webbed feet},
  parent={animal}
}
```

The previous example did this by loading both the `terms.bib` and `animals.bib` files and aliasing the custom `identifier` field to `parent`. In this example, the custom field is aliased to `type`, which effectively makes the definitions behave like:

```
@index{animal}
@entry{duck, name={duck},
  description={a waterbird with webbed feet},
  type={animal}
}
```

The aim here is for the `animal` entry to be placed in the main glossary so that it’s listed with \printunsrtglossary. The `duck` entry is placed in a glossary that has a label (`animal`) that matches the label of the “parent” entry (even though it’s technically not a parent). This new glossary (`animal`) can be automatically defined by invoking `bib2gls` with the `--provide =glossaries` switch.

This example document defines a custom handler function that will do the current entry as normal (with \glsxtrunsrtdo) but will then check for the existence of a glossary that has the same label as the current entry. This requires the starred version of \ifglossaryexists to included ignored glossaries in the existence check. If the glossary exists, it’s then displayed using \printunsrtinnerglossary:

```
\newcommand{\nestedhandler}[1]{
  \glsxtrunsrtdo{#1}%
  \ifglossaryexists*{#1}%
  {%
    \printunsrtinnerglossary[type={#1},leveloffset={++1},groups={false}]
    {}{}
  }%
  {}
}
```
The `leveloffset` option is required to achieve a hierarchical effect (provided the glossary style supports it) and the `groups={false}` option is needed to prevent letter groups showing for the nested glossary, which would otherwise create a strange effect. (This example uses the treegroup style, which provides a hierarchical glossary with letter groups.)

The \printunsrtglossary handler macro then needs to be set to this custom macro when the main glossary is displayed:

\printunsrtglossary*{\let\printunsrtglossaryhandler\nestedhandler}

The main difficulty comes with ensuring that all the necessary entries are selected. Now that the custom `identifier` field has been aliased to `type` rather than `parent`, the `animal` entry is no longer considered a dependent. The `duck` entry has been referenced in the document with `\gls` but the `animal` entry hasn’t. The previous example ensured that the `animal` entry was selected because it was a parent of a selected entry. If the same resource options are used in this example, the main glossary will be empty, which means that the nested glossaries won’t be displayed either.

One way to ensure that the `animal`, `mineral` and `vegetable` entries are selected is to identify the `type` field as a dependency field:

\GlsXtrLoadResources[src={terms,animals,minerals,vegetables},
  field-aliases={identifier=type},
  dependency-fields={type}]

This will achieve the same effect as the `sample-hierarchical.tex` document, but it’s a far more convoluted method. The reason this example document is listed here is to demonstrate a slightly modified hierarchical effect that can’t be achieved through the normal method.

Suppose that, for some strange reason, I want the “animal”, “mineral” and “vegetable” entries to be listed in a different order (say, reverse alphabetical). The other entries (“duck” etc) need to be sorted in normal alphabetical order.

The `sort` option applies the same sort method to all hierarchical levels. The sort value chosen for particular entries can be altered through the use of fallbacks (such as the `entry-sort-fallback` or `symbol-sort-fallback` options) and a letter comparator may be used to resolve identical sort values (`identical-sort-action`), but the same sort algorithm is applied to all entries in the same set (primary, secondary or dual within the same resource set). The only way to apply different sort methods is to separate the entries into different resource sets (or use dual or secondary sorting).

This can be achieved by having one resource set for the main entries with one sort method and another resource set for all the other entries with a different sort method:

\GlsXtrLoadResources[src={terms},sort={en-reverse}]
\GlsXtrLoadResources[src={animals,minerals,vegetables},sort={en},
  field-aliases={identifier=type},dependency-fields={type}]

This works when cross-resource dependencies are permitted (see section 1.4). In the event that cross-resource dependencies aren’t permitted, the selection criteria is more complicated:
\GlsXtrLoadResources[src={terms,animals,minerals,vegetables},
  sort={en-reverse},
  field-aliases={identifier=parent},
  selection={ancestors but not recorded}]
\GlsXtrLoadResources[
  src={animals,minerals,vegetables},
  field-aliases={identifier=type},
  dependency-fields={type}
  sort={en}]

Fortunately in this example, cross-resource dependencies are permitted so the simpler alternative works. (If they’re not permitted, the \texttt{bib2gls} transcript file will contain “Cross-resource references can’t be supported for resource set \textit{⟨filename⟩}.”)

The complete code is listed below. The document build is:

\pdflatex sample-nested
\bib2gls --group --provide-glossaries sample-nested
\pdflatex sample-nested

The complete document is shown in figure 8.2.

\documentclass[12pt,a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage[record, % use bib2gls
  nostyles, % don’t load default styles
  postdot, % add a full stop after the description
  % load glossary-tree.sty and patch styles:
  % stylemods={tree},
  style=treegroup]{glossaries-extra}
\GlsXtrLoadResources[src={terms},sort={en-reverse}]
\GlsXtrLoadResources[
  src={animals,minerals,vegetables},
  field-aliases={identifier=type},
  dependency-fields={type},
  sort={en}]

\newcommand{\nestedhandler}[1]{% 
  \glsxtrunsrtdo{#1}%
  \% Is there a glossary whose label (type) matches this entry’s label?

454
\begin{document}

Some sample terms: \gls{duck}, \gls{sealion}, \gls{armadillo}, \gls{seal}, \gls{aardvark}, \gls{amethyst}, \gls{aquamarine}, \gls{diamond}, \gls{dolomite}, \gls{chalcedony}, \gls{citrine}, \gls{quetzalcoatlite}, \gls{cabbage}, \gls{cauliflower}, \gls{artichoke}, \gls{courgette}.

\GlsXtrSetDefaultNumberFormat{glsignore}% ignore records in the glossary

\end{document}

\textbf{sample-constants.tex}

This example uses the constants.bib file. The aim here is to just have a list of all the constants defined in the .bib file. (There are no references in the document.) This means I need to use:

\texttt{selection={all}}

in order to select all entries. I also need to alias the custom \texttt{@constant} entry type otherwise all the entries will be ignored. I decided to make \texttt{@constant} behave like \texttt{@number} for semantic reasons:

\texttt{entry-type-aliases={constant=number}}

The custom fields also need aliasing:

\texttt{field-aliases={}
    identifier=category,
    constantsymbol=name,
    constantname=description,
    value=user1,
    definition=user2,
    alternative=user3,
}\texttt{}}
Some sample terms: duck, sea lion, armadillo, seal, aardvark, amethyst, aquamarine, diamond, dolomite, chalcedony, citrine, quetzalcoatlite, cabbage, cauliflower, artichoke, courgette.

Glossary

V
vegetable
artichoke a variety of thistle cultivated as food. 1
cabbage vegetable with thick green or purple leaves. 1
cauliflower type of cabbage with edible white flower head. 1
courgette immature fruit of a vegetable marrow. 1

M
mineral
amethyst purple variety of quartz. 1
aquamarine light blue variety of beryl. 1
beryl composed of beryllium aluminium cyclosilicate.
chalcedony cryptocrystalline variety of quartz. 1
citrine yellow variety of quartz. 1
diamond a metastable allotrope of carbon. 1
dolomite an anhydrous carbonate mineral. 1
quartz hard mineral consisting of silica.
quartz amethyst a rare tellurium oxysalt mineral. 1

A
animal
aardvark nocturnal African burrowing mammal. 1
armadillo nocturnal insectivore with large claws. 1
duck a waterbird with webbed feet. 1
sea lion a large type of seal. 1
seal sea-dwelling fish-eating mammal with flippers. 1
I decided to use the altlist style, so I’ve instructed bib2gls to determine the widest name:

\texttt{set-widest}

It’s always a good idea to specify the glossary type when using \texttt{set-widest}, although in this example there’s only one glossary so it doesn’t make much difference.

\texttt{type=\{main\}}

I decided to order the constants according to their (approximate) numerical value. I’ve aliased the custom \texttt{value} field to \texttt{user1}, so I can sort by that field using a numerical comparison:

\texttt{sort-field=\{user1\},}
\texttt{sort=\{double\}}

There are three entries without the \texttt{user1} field (as the custom \texttt{value} field is missing in the .bib file): zero, one, and imaginary. In the case of zero and one the exact value can be obtained from the \texttt{name} field. Since I’ve change the default \texttt{sort-field}, I can’t use \texttt{symbol-sort-fallback}. Instead I need to use:

\texttt{missing-sort-fallback=\{name\}}

What happens with the imaginary entry? It has no real representation. The transcript (.glg) file shows the message:

\texttt{Warning: Can't parse sort value 'i' for: imaginary}

With the numerical sort methods, if the field can’t be parsed the value defaults to 0. This means that both zero and imaginary have 0 as the sort value, so the \texttt{identical-sort-action} is implemented. The default setting means that bib2gls will fallback on comparing the entry labels, so imaginary comes before zero.

Since I’m just using the alttree style, I only need glossary-tree. I can improve efficiency in the document build by preventing the other glossary style packages from being loaded using the \texttt{nostyles} package option. This also prevents glossary-tree from being loaded, but I can both load it and patch the styles with glossaries-extra-stylemods through the option \texttt{stylemods=\{tree\}}. Since the default list style is no longer available, I need to set a new default with \texttt{style=\{alttree\}}. I also want to automatically insert a full stop after the description, which can be done with \texttt{postdot}. Don’t forget that the \texttt{record} option is always needed when using bib2gls. This means that the glossaries-extra package needs to be loaded as follows:

\texttt{\usepackage[record,nostyles,postdot,stylemods=\{tree\},style=\{alttree\}]{glossaries-extra}}

I’ve assigned the custom \texttt{constantname} field to the \texttt{description} field and the custom \texttt{constantsymbol} field to the \texttt{name} field. This means that by default the glossary list will just show the symbolic representation and the constant’s name. I’d like to append the value and definition after the description. With the base glossaries package this would require
defining a new glossary style but with glossaries-extra it can easily be achieved through the post-description hook.

I’ve aliased the custom identifier field to category, which means that all the entries will have the category set to constant. The post-description hook is obtained from \glsxtrpostdesc\langle category\rangle, so I need to define the command \glsxtrpostdescconstant. A simple definition is:

\newcommand{\glsxtrpostdescconstant}{%
    \space (approximately \glsentryuseri{\glscurrententrylabel})%
    : \glsentryuserii{\glscurrententrylabel}%
}%

This is fine if all entries have the user1 and user2 fields set. A more generic approach tests for the existence of these fields. This can either be done with \ifglshasfield:

\newcommand{\glsxtrpostdescconstant}{%
    \ifglshasfield{user1}{\glscurrententrylabel}{ (approximately \glscurrentfieldvalue)}%
    {}%
    \ifglshasfield{user2}{\glscurrententrylabel}{: \glscurrentfieldvalue}%
    {}%
}%

or with \glsxtrifhasfield:

\newcommand{\glsxtrpostdescconstant}{%
    \glsxtrifhasfield{useri}{\glscurrententrylabel}{ (approximately \glscurrentfieldvalue)}%
    {}%
    \glsxtrifhasfield{userii}{\glscurrententrylabel}{: \glscurrentfieldvalue}%
    {}%
}%

(Note the need to use the internal field label useri and userii with \glsxtrifhasfield.)

A modification can be made to also show the alternative representation (obtained from the custom alternative field which has been aliased to user3):

\newcommand{\glsxtrpostdescconstant}{%
    \glsxtrifhasfield{useriii}{\glscurrententrylabel}{ (also denoted \glscurrentfieldvalue
    \glsxtrifhasfield{useri}{\glscurrententrylabel}{, approximately \glscurrentfieldvalue}%
    {}%
    )%
}%

}
8 Examples: sample-constants.tex

\begin{verbatim}
{\glxtrifhasfield{useri}{\glscurrententrylabel}\%
{ (approximately \glscurrentfieldvalue)}%
{}}%
\glxtrifhasfield{userii}{\glscurrententrylabel}\%
{: \glscurrentfieldvalue}%
{}}%

If you have at least glossaries-extra v1.31, it’s better to use:

\glsdefpostdesc{constant}

instead of:

\newcommand{\glsxtrpostdescconstant}

as it can guard against accidental misspelling of the glsxtrpostdesc part of the command name.

The complete code is listed below. The document build is:

deplatex sample-constants
bib2gls sample-constants
deplatex sample-constants

The complete document is shown in figure 8.3.

documentclass[12pt,a4paper]{article}

\usepackage[T1]{fontenc}
\usepackage{upgreek}
\usepackage[record, % use bib2gls
nostyles, % don't load default styles
postdot, % add dot after descriptions
% load glossary-tree.sty and patch styles:
stylemods={tree},
style=alttree]{glossaries-extra}

\GlsXtrLoadResources[ 
src={constants}, % data in constants.bib
% make @constant behave like @number
entry-type-aliases={constant=number},
field-aliases={ 
    identifier=category,
    constantsymbol=name,
    constantname=description,
    value=user1,
    459
}
8 Examples: sample-chemical.tex

definition=user2,
alternative=user3
},
type=main,
set-widest,
sort-field=user1,
missing-sort-fallback=name,
sort=double,
selection=all
]

\newcommand{\glsxtrpostdescconstant}{%
\glsxtrifhasfield{useriii}{\glscurrententrylabel}%
{ (also denoted \glscurrentfieldvalue
  \glsxtrifhasfield{useri}{\glscurrententrylabel}%
  {, approximately \glscurrentfieldvalue}%
  {}%}
)%
}%
{%
\glsxtrifhasfield{useri}{\glscurrententrylabel}%
{ (approximately \glscurrentfieldvalue)%
  {}%}
\glsxtrifhasfield{userii}{\glscurrententrylabel}%
{ \glscurrentfieldvalue}%
}{%}

\begin{document}
\printunsrtglossary[title={Constants}]
\end{document}

sample-chemical.tex

This example just uses the chemicalformula.bib file. The aim here is to have a list of chemical formulae referenced in the document but not have a number list. I could use the nonumberlist package option to suppress the number list display, but it’s more efficient to instruct bib2gls to not save the number list with:

\texttt{save-locations={false}}

All entries are defined in chemicalformula.bib using a custom entry type \texttt{@chemical} which needs to be aliased in order for the entries to be recognised:

\texttt{entry-type-aliases={chemical=symbol}}
Constants

- $i$: imaginary unit (also denoted $j$): defined as $i^2 = -1$.
- $0$: zero: nothing or nil.
- $\gamma$: Euler’s constant (approximately 0.57721): the limit of
  \[
  \sum_{n=1}^{\infty} \frac{1}{n} - \ln n
  \]
  as $n \to \infty$.
- $1$: one: single entity, unity.
- $\zeta(3)$: Apéry’s constant (approximately 1.2020569): a special value of the Riemann zeta function.
- $\lambda$: Conway’s constant (approximately 1.30357): the invariant growth rate of all derived strings.
- $\sqrt{2}$: Pythagoras’ constant (approximately 1.41421): the square root of 2.
- $\phi$: golden ratio (approximately 1.61803): the ratio $\frac{1 + \sqrt{5}}{2}$.
- $e$: Euler’s number (approximately 2.71828): base of natural logarithms.
- $\pi$: pi (approximately 3.14159): the ratio of the length of the circumference of a circle to its diameter.

Figure 8.3: sample-constants.pdf
Additionally, the entries only have custom fields, so these also need to be aliased. In this case I want the formula in the `name` field and the chemical name in the `description` field:

```
field-aliases={formula=name,chemicalname=description}
```

The `@symbol` entry type falls back on the label for the `sort` value by default, but I’ve decided to fallback on the `name` field for sorting:

```
symbol-sort-fallback={name}
```

An alternative approach would simply be to alias `@chemical` to `@entry` instead.

Since the `name` field contains chemical formulae rather than words, it makes more sense to use one of the letter sort methods rather than a locale collator. In this case the names contain mixtures of letters and numbers, so one of the letter-number sort methods (listed in table 5.4) would be appropriate.

I want to use the alttreegroup style (provided by glossary-tree). Since I don’t require the other style packages, I’ve used `nostyles` to suppress the automatic loading and `stylemods ={tree}` to both load glossary-tree and patch it. The alttreegroup style needs to know the widest name, so I’ve use `set-widest` for convenience. The default behaviour of the tree styles is to format the name in bold. This is done through the command \glstreenamefmt which is defined as:

```
\newcommand*{\glstreenamefmt}[1]{\textbf{#1}}
```

The group headings use `\glstreegroupheaderfmt` which defaults to `\glstreenamefmt`. Since I want to keep bold headings, I need to redefine this as well:

```
\renewcommand*{\glstreenamefmt}[1]{#1}
\renewcommand*{\glstreegroupheaderfmt}[1]{\textbf{#1}}
```

(For a more compact layout, you could use `mcolalttreegroup` instead.) I also need the `--group` switch to make the sort method automatically assign letter groups.

The complete code is listed below. The document build is:

```
pdflatex sample-chemical
bib2gls --group sample-chemical
pdflatex sample-chemical
```

The complete document is shown in figure 8.4.

```
\documentclass[a4paper]{article}

\usepackage[T1]{fontenc}
\usepackage[version=4]{mhchem}
\usepackage[record, % use bib2gls
nostyles,% don't load default styles
stylemods={tree},% load glossary-tree and patch styles
style=alttreegroup]{glossaries-extra}
```

462
8 Examples: sample-bacteria.tex

\GlsXtrLoadResources[  
src={chemicalformula}, % definitions in chemicalformula.bib  
entry-type-aliases={chemical=symbol},  
field-aliases={formula=name, chemicalname=description},  
symbol-sort-fallback=name, % use name field as fallback for sort  
sort=letternumber-case, % case-sensitive letter-number sort  
set-widest, % needed for alttree styles  
save-locations=false % don't create location lists  ]

\renewcommand*{\glstreefmt}[1]{#1}
\renewcommand*{\glstreegroupheaderfmt}[1]{\textbf{#1}}

\begin{document}
\section{Sample}
Reference Entries: \gls{Al2SO43}, \gls{H2O}, \gls{C6H12O6}, \gls{CH3CH2OH}, \gls{CH2O}, \gls{OF2}, \gls{O2F2}, \gls{SO42-}, \gls{H3O+}, \gls{OH-}, \gls{O2}, \gls{AlF3}, \gls{O}, \gls{Al2CoO4}, \gls{As4S4}, \gls{C10H10O4}, \gls{C5H4NCOOH}, \gls{C8H10N4O2}, \gls{SO2}, \gls{S2O72-}, \gls{SbBr3}, \gls{Sc2O3}, \gls{Zr3PO44}, \gls{ZnF2}.
\printunsrtglossary
\end{document}

sample-bacteria.tex

This example just uses the bacteria.bib file. The aim here is to have a simple list of the bacteria referenced in the document. Bacteria names are often shown in the long form on first use (without the short form) and then the short form on subsequent use. This can easily be done with the long-only-short-only style. Bacteria are usually typeset in italic. It’s best to create a semantic command for this:

\newcommand{\bacteriafont}[1]{\emph{#1}}

There are two methods to apply this to the bacteria entries. The first is to redefine the formatting commands used by the long-only-short-only style:

\renewcommand*{\glsabbrvonlyfont}[1]{\bacteriafont{#1}}
\renewcommand*{\glslongonlyfont}[1]{\bacteriafont{#1}}

This is fine if I don’t intend to use this style for other types of abbreviations. However, I may decide to extend the document at a later date to include other abbreviations that need long-only-short-only but shouldn’t be emphasized. This can be done through the use of category
1 Sample

Reference Entries: Al₂(SO₄)₃, H₂O, C₆H₁₂O₆, CH₃CH₂OH, CH₂O, OF₂, O₂F₂,
SO₄²⁻, H₂O⁺, OH⁻, O₂, AlF₃, O, Al₃C₃O₃, As₄S₄, C₃H₁₀O₄, C₅H₄NCOOH,
C₆H₁₃N₂O₂, SO₂, S₂O₇²⁻, SbBr₃, Sc₂O₃, Zr₃(PO₄)₄, ZnF₂.

Glossary

A
AlF₃ aluminium trifluoride
Al₂(SO₄)₃ aluminium sulfate
Al₂CoO₄ cobalt blue
As₄S₄ tetraarsenic tetra sulfide

C
CH₂O formaldehyde
CH₃CH₂OH ethanol
C₃H₃NCOOH niacin
C₆H₁₂O₆ glucose
C₈H₁₀N₄O₂ caffeine
C₁₀H₁₀O₄ ferulic acid

H
H₂O water
H₃O⁺ hydronium

O
O oxygen
OF₂ oxygen difluoride
OH⁻ hydroxide ion
O₂ dioxygen
O₂F₂ dioxygen difluoride

S
SO₂ sulfur dioxide
SO₄²⁻ sulfate
S₂O₇²⁻ disulfate ion
SbBr₃ antimony(III) bromide
Sc₂O₃ scandium oxide

Z
ZnF₂ zinc fluoride
Zr₃(PO₄)₄ zirconium phosphate

Figure 8.4: sample-chemical.pdf
attributes. The font used for the \textit{name} in the glossary is governed by the glossnamefont attribute, the font used for the \textit{description} in the glossary is governed by the glossdescfont attribute and the font used by commands like \texttt{\gls} in the document is governed by the textformat attribute (glossaries-extra v1.21+). So if I set the category to bacteria then I can do:

\setabbreviationstyle[bacteria]{long-only-short-only} \\
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont} \\
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont}

and (if the \textit{description} field is displayed in the glossary):

\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont}

(Note that the attribute value is the control sequence name without the initial backslash.)

I’d like to use the bookindex style, which is provided by the glossary-bookindex package. This isn’t loaded automatically, but it can be loaded through the stylemods package option:

\usetikzlibrary{glossaries-extra}
\usepackage[record,% use bib2gls
nostyles,% don't load default style packages
stylemods={bookindex},% load glossary-bookindex.sty and patch styles
style={bookindex}]{glossaries-extra}

I’ve used the nostyles package option to suppress loading the default style packages, since I’m not using them. If you inspect the .log file, you may notice that glossary-tree is still loaded. This is because it’s required by glossary-bookindex as the bookindex style is based on the index style provided by glossary-tree. With this style I need to use the --group switch to instruct the sort method to automatically create the letter groups.

The bookindex style doesn’t show the \textit{description} field (which means I don’t need the glossdescfont attribute) and, since the long-only-short-only style sets the \textit{name} to the short form by default, only the short form will show in the glossary. I’d rather it was just the long form. This could simply be done using replicate-fields to copy the long field to the name field:

replicate-fields={long=name}

Again, I want to consider the possibility of adding other types of abbreviations and this might not be appropriate for them (for example, I might want some abbreviations with the long form followed by the short form in parentheses). Another approach is to redefine \texttt{\glsxtrbookindexname} which is used by the bookindex style to display the name. This takes the entry’s label as the argument. The default definition is:

\newcommand*{\glsxtrbookindexname}[1]{\glossentryname[#1]}

This can be changed to test for the entry’s category:

\glsxtrbookindexname[1]\glossentryname[#1]

\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont}

1glossary-bookindex is distributed with glossaries-extra v1.21+. 
8 Examples: sample-bacteria.tex

\renewcommand*{\glsxtrbookindexname}{%\glssifcategory{#1}{bacteria}\{%\glossentrynameother{#1}{long}\}\{%\glossentryname{#1}\}\}%}

Note that I've used \glossentrynameother here rather than \glsentrylong. This ensures that it follows the same formatting as \glossentryname (so it will use \glsnamefont or the glossnamefont attribute, the glossname attribute, and the post-name hook, if set). In this case it picks up the glossnamefont attribute, which is used instead of \glsnamefont.

If the sort field is missing for abbreviation styles, the fallback value is the short field (not the name field). In this case it would be better to fallback on the long field instead, which can be done with the abbreviation-sort-fallback option:

abbreviation-sort-fallback={long}

If I do add other types of abbreviations, they will all be sorted according to the long form, but at least this way I can have some (long) ((short)) names as well.

The complete code is listed below. The document build is:

pdflatex sample-bacteria
bib2gls --group sample-bacteria
pdflatex sample-bacteria

This simple example only references entries on the first page so all entries just have 1 in the number list. The complete document is shown in figure 8.5.

\documentclass[12pt,a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[record,% use bib2gls
nostyles,% don't load default styles
% load glossary-bookindex.sty and patch styles:
stylemods={bookindex},
style=bookindex]{glossaries-extra}

% abbreviation style must be set before \GlsXtrLoadResources
\setabbreviationstyle[bacteria]{long-only-short-only}
\GlsXtrLoadResources[
src=bacteria,% data in bacteria.bib
category=bacteria,
abbreviation-sort-fallback=long
]
\newcommand{\bacteriafont}{\emph{\#1}}
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont}

\renewcommand*{\glsxtrbookindexname}[1]{%
  \glsifcategory{#1}{bacteria}
  {\glossentrynameother{#1}{long}}%
  {\glossentryname{#1}}%
}%

\begin{document}
\section{First Use}
\gls{cbotulinum}, \gls{pputida}, \gls{cperfringens}, \gls{bsubtilis}, \gls{ctetani}, \gls{pcomposti}, \gls{pfimeticola}, \gls{cburnetii}, \gls{raustralis}, \gls{rrickettsii}.

\section{Next Use}
\gls{cbotulinum}, \gls{pputida}, \gls{cperfringens}, \gls{bsubtilis}, \gls{ctetani}, \gls{pcomposti}, \gls{pfimeticola}, \gls{cburnetii}, \gls{raustralis}, \gls{rrickettsii}.

\printunsrtglossary[title={Bacteria Index}]
\end{document}

\textbf{sample-units1.tex}

This example uses the baseunits.bib and derivedunits.bib files. The aim here is to have a glossary in two blocks: base units and derived units. This can be achieved by first loading baseunits.bib with group set to the desired group title ("Base Units" in this case) and then load derivedunits.bib with the group set to the desired title ("Derived Units" in this case). Remember that the group field needs to be used as a label. If the group title contains any problematic characters or commands, then it’s better to use labels:

\texttt{group={baseunits}}

for the first resource set and

\texttt{group={derivedunits}}

for the second, and then set the group titles:

\texttt{\glsxtrsetgrouptitle{baseunits}{Base Units}}
\texttt{\glsxtrsetgrouptitle{derivedunits}{Derived Units}}
1 First Use

Clostridium botulinum, Pseudomonas putida, Clostridium perfringens, Bacillus subtilis, Clostridium tetani, Planifilum composti, Planifilum fimeticola, Coxiella burnetii, Rickettsia australis, Rickettsia rickettsii.

2 Next Use


Bacteria Index

<table>
<thead>
<tr>
<th>B</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacillus subtilis, 1</td>
<td>Planifilum composti, 1</td>
</tr>
<tr>
<td>Clostridium botulinum, 1</td>
<td>Planifilum fimeticola, 1</td>
</tr>
<tr>
<td>Clostridium perfringens, 1</td>
<td>Pseudomonas putida, 1</td>
</tr>
<tr>
<td>Clostridium tetani, 1</td>
<td>R</td>
</tr>
<tr>
<td>Coxiella burnetii, 1</td>
<td>Rickettsia australis, 1</td>
</tr>
<tr>
<td></td>
<td>Rickettsia rickettsii, 1</td>
</tr>
</tbody>
</table>

Figure 8.5: sample-bacteria.pdf
I’ve used this method to make it easier to adapt to other languages that may need extended characters in the group titles. The \texttt{group} option requires the \texttt{--group} switch to ensure that the \texttt{group} field is correctly assigned.

The \texttt{baseunits.bib} file uses a custom entry type \texttt{@unit}, which must be aliased otherwise \texttt{bib2gls} will ignore the entries. I decided to use \texttt{@symbol} for semantic reasons:

\begin{verbatim}
entry-type-aliases={unit=symbol}
\end{verbatim}

Similarly for the custom \texttt{@measurement} entry type in \texttt{derivedunits.bib}:

\begin{verbatim}
entry-type-aliases={measurement=symbol}
\end{verbatim}

Remember that \texttt{@symbol} uses the label as the default sort fallback, so I’ve changed it to use \texttt{name} instead:

\begin{verbatim}
symbol-sort-fallback={name}
\end{verbatim}

An alternative approach would be to alias \texttt{@unit} and \texttt{@measurement} to \texttt{@entry} instead.

Since there’s no \texttt{type} set, all entries end up in the main glossary, but since there are two resource commands the glossary ends up with sorted blocks.

The document doesn’t include any commands like \texttt{\gls}, so I’ve use \texttt{selection={all}} to select all entries in the .bib files. There won’t be any number lists since there are no records. I need a glossary style that shows the \texttt{symbol} field so I’ve used \texttt{mcolindexgroup}. Again I’ve suppressed the automatic loading of the default styles with \texttt{nostyles} and used \texttt{stylemods={mcols}} to load \texttt{glossary-mcols} and patch the styles. Note that although I’ve used \texttt{nostyles}, the glossary–tree style is loaded as it’s required by \texttt{glossary-mcols}.

As with the previous example, the custom fields need to be aliased:

\begin{verbatim}
field-aliases=
    {unitname=name, 
     unitsymbol=symbol, 
     measurement=description
}
\end{verbatim}

The complete document code is listed below. The document build is:

\begin{verbatim}
pdflatex sample-units1 
bib2gls --group sample-units1 
pdflatex sample-units1
\end{verbatim}

The complete document is shown in figure 8.6.
8 Examples: sample-units2.tex

```
style=mcolindexgroup]{glossaries-extra}

\GlsXtrLoadResources[
  src={baseunits},
  % make @unit act like @symbol:
  entry-type-aliases={unit=symbol},
  field-aliases={
    unitname=name,
    unitsymbol=symbol,
    measurement=description
  },
  symbol-sort-fallback=name,
  selection={all},
  group={baseunits}
]

\GlsXtrLoadResources[
  src={derivedunits},
  % make @measurement act like @symbol:
  entry-type-aliases={measurement=symbol},
  field-aliases={
    unitname=name,
    unitsymbol=symbol,
    measurement=description
  },
  symbol-sort-fallback=name,
  selection={all},
  group={derivedunits}
]

\glsxtrsetgrouptitle{baseunits}{Base Units}
\glsxtrsetgrouptitle{derivedunits}{Derived Units}

\begin{document}

\printunsrtglossaries

\end{document}
```

**sample-units2.tex**

This example is provided for comparison with sample-units1.tex. Instead of having a single glossary with sorted blocks this example has two glossaries:

```
\newglossary*{baseunits}{Base Units}
\newglossary*{derivedunits}{Derived Units}
```
Glossary

Base Units
- ampere (A) electric current
- candela (cd) luminous intensity
- kelvin (K) thermodynamic temperature
- kilogram (kg) mass
- metre (m) length
- mole (mol) amount of substance
- second (s) time

Derived Units
- ampere per square metre (A m\(^{-2}\)) density
- candela per square metre (cd m\(^{-2}\)) luminance
- cubic metre (m\(^3\)) volume
- cubic metre per kilogram (m\(^3\) kg\(^{-1}\)) specific volume
- metre per second (m s\(^{-1}\)) velocity
- metre per second squared (m s\(^{-2}\)) acceleration
- mole per cubic metre (mol m\(^{-3}\)) concentration
- per metre (m\(^{-1}\)) wave number
- square metre (m\(^2\)) area
I’ve used the `section` package option to use `\section*` for the glossary titles. This overrides the default `\chapter*` which is used with book or report type of classes. I’ve also used the `nomain` option to suppress the creation of the main glossary as I want to define my own glossary types instead.

As before the custom entry types need to be aliased:

```latex
entry-type-aliases={unit=symbol}
```

for the first resource set and

```latex
entry-type-aliases={measurement=symbol}
```

for the second. Similarly for the custom entry fields:

```latex
field-aliases=
  
  unitname=name,
  unitsymbol=symbol,
  measurement=description
}
```

The `--group` switch is needed to ensure that the `group` field is automatically assigned by the sort method.

The complete document code is listed below. The document build is:

```bash
pdflatex sample-units2
bib2gls --group sample-units2
pdflatex sample-units2
```

The complete document is shown in figure 8.7.

```latex
\documentclass[a4paper]{report}
\usepackage{siunitx}
\usepackage[record,% use bib2gls
  nomain,% don't define 'main' glossary
  section,% use \section* for glossary headings
  nostyles,% don't load default styles
  stylemods={mcols},% load glossary-mcols.sty and patch
  style=mcolindex]{glossaries-extra}
\newglossary*[baseunits]{Base Units}
\newglossary*[derivedunits]{Derived Units}
\GlsXtrLoadResources[
  src={baseunits},
  type=baseunits,
  % make @unit act like @symbol:
  entry-type-aliases={unit=symbol},
```
8 Examples: sample-units3.tex

field-aliases=
    unitname=name,
    unitsymbol=symbol,
    measurement=description
],
symbol-sort-fallback=name,
selection={all}
]
\GlsXtrLoadResources[
    src={derivedunits},
    type=derivedunits,
    % make @measurement act like @symbol:
    entry-type-aliases={measurement=symbol},
    field-aliases=
        unitname=name,
        unitsymbol=symbol,
        measurement=description
],
symbol-sort-fallback=name,
selection={all}
]
\begin{document}
\chapter*{Glossaries}
\printunsrtglossary[type=baseunits,nogroupskip]
\printunsrtglossary[type=derivedunits,style=indexgroup]
\end{document}

sample-units3.tex

This is another example that uses the baseunits.bib and derivedunits.bib files. As before the custom fields need to be aliased:

field-aliases=
    unitname=name,
    unitsymbol=symbol,
    measurement=description
}

This time I want two glossaries containing all the units (base and derived) where the first glossary is ordered by name and the second is ordered by symbol. This can be done with a single resource command that instructs bib2gls to make the custom @unit and @measurement entry types behave like @dualsymbol:
Glossaries

Base Units

ampere (A) electric current

candela (cd) luminous intensity

kelvin (K) thermodynamic temperature

kilogram (kg) mass

metre (m) length

mole (mol) amount of substance

second (s) time

Derived Units

A

ampere per square metre (A m⁻²) density

C

candela per square metre (cd m⁻²) luminance

cubic metre (m³) volume

cubic metre per kilogram (m³ kg⁻¹) specific volume

M

metre per second (m s⁻¹) velocity

metre per second squared (m s⁻²) acceleration

mole per cubic metre (mol m⁻³) concentration

P

per metre (m⁻¹) wave number

S

square metre (m²) area

Figure 8.7: sample-units2.pdf
entry-type-aliases={
    unit=dualsymbol,
    measurement=dualsymbol
}

This causes the name and symbol fields to be swapped in the dual list. Remember that the fallback for the sort field is the label for the symbol entry types so I need symbol-sort-fallback={name} to fallback on name field instead. (Alternative, I could just sort by the name field instead using sort-field={name}.)

The primary entries can still be sorted according to the default locale collator, but the dual entries need a sort method that’s better suited to symbols. Fortunately, bib2gls has some (very limited) support for siunitx and is able to interpret the \si commands in the sample .bib files. Since si units are a mix of letters and numbers I’ve used one of the letter-number methods listed in table 5.4.

I’ve decided to define a custom style for the first glossary. Since it’s based on the long3col-booktabs style I need to load glossary-longbooktabs, which can conveniently be done with the stylemods option. This uses longtable (provided by longtable, which is automatically loaded) which means an extra \LaTeX{} call is required in the build process to ensure the column widths are correct. Again I’m using nostyles to suppress the automatic loading of the default styles, however glossary-tree will be loaded as it’s listed in the value of stylemods and glossary-long will be loaded as it’s required by glossary-longbooktabs. I can’t use my custom style in the style package option as it hasn’t been defined at that point. The default list style is now unavailable since nostyles has prevented it from being defined, so I’ve used style={alttree} to ensure there’s a valid default style.

Since my custom style is based on one of the long styles, I need to set the length register \glsdescwidth to adjust the width of the description column:

\setlength{\glsdescwidth}{.4\hsize}

The long3col-booktabs style sets up a three column longtable so I just need to adjust the table header (to rename the column headers) and the way each row is formatted:

\newglossarystyle{units}{% style name
    \setglossarystyle{long3col-booktabs}%
    \renewcommand*{\glossaryheader}{%
        \toprule
        \bfseries Name & \bfseries Measurement & \bfseries Symbol
        \midrule
    }%
    % main entries:
    \renewcommand{\glossentry}[2]{% \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
There are no sub-entries in this document so I haven’t bothered to redefine \subglossentry. (The tabular styles aren’t appropriate for hierarchical glossaries.) This puts the symbol into the third column (rather than the location list, which is ignored). This style supports the letter group separator (although it doesn’t title the groups), so if I want this I need to use the --group switch.

I also need to make sure I’ve defined a glossary for the dual entries:

\newglossary*{units}{Units of Measurement (by SI unit)}

and specify the glossary types for the primary and dual entries:

type={main},
dual-type={units}

The complete document code is listed below. The document build is:

pdflatex sample-units3
bib2gls --group sample-units3
pdflatex sample-units3
pdflatex sample-units3

The two pages of the document are shown in figure 8.8.

\documentclass[12pt,a4paper]{report}

\usepackage{siunitx}
\usepackage[record,% use bib2gls
    nostreams,% don't load default styles
    % load glossary-tree.sty and glossary-longbooktabs.sty and patch:
    stylemods={tree,longbooktabs},
    style=alttree]{glossaries-extra}

\newglossary*{units}{Units of Measurement (by SI unit)}

\GlsXtrLoadResources[
    % data in baseunits.bib and derivedunits.bib:
    src={baseunits,derivedunits},
    field-aliases={
        unitname=name,
        unitsymbol=symbol,
        measurement=description
    },
    symbol-sort-fallback={name},
]
8 Examples: sample-units3.tex

\begin{verbatim}
selection=all,% select all entries
% make @measurement and @unit act like @dualsymbol:
entry-type-aliases={
    measurement=dualsymbol,
    unit=dualsymbol,
},
set-widest,% needed for alttree style
dual-sort={letternumber-upperlower},
type=main,% put primary entries in 'main' glossary
dual-type={units} % put dual entries in 'units' glossary
\]
\setlength{\glsdescwidth}{.4\hsize}

% define custom glossary style
\newglossarystyle{units}{% base it on long3col-booktabs
    \setglossarystyle{long3col-booktabs}%
    \renewcommand*{\glossaryheader}{%
        \toprule
        \bfseries Name &
        \bfseries Measurement &
        \bfseries Symbol
        \bottomrule\endfoot
    }%
% main entries:
    \renewcommand{\glossentry}[2]{%
        \glossentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
        \glossentrydesc{##1}\glspostdescription &
        \glossentrysymbol{##1}\tabularnewline
    }%
}\begin{document}
\printunsrtglossary[title={SI Units of Measurement},
    style={units}]
\printunsrtglossary[type=units]
\end{document}
\end{verbatim}

This example uses the sample files books.bib, films.bib, no-interpret-preamble.bib and interpret-preamble.bib. The aim is to produce a combined list of books and films in a single glossary. The films are based on some of the books so some of the entries have the same name. The default setting for identical sort values is \texttt{identical-sort-action={id}}, which means that the ordering for the duplicate names is based on the entry labels. This can lead to the odd effect of sometimes having the film listed first (\texttt{film.thebigsleepl comes before thebigsleep}) and sometimes having the book listed first (\texttt{brightonrock comes before film.brightonrock}).

One possible solution would be to also assign prefixes for the book labels, but \texttt{label-prefix} is applied to all primary entries for the given resource set and can’t be applied selectively, so this would require editing the books.bib file.

A more consistent approach would be to fallback on the category. This means that the \texttt{category} field needs to be set. There are two simple ways to achieve this: use \texttt{category=\{same as base\}} (which sets the \texttt{category} to \texttt{books} for entries in books.bib and to \texttt{films} for entries in films.bib) or alias the custom \texttt{identifier} field to \texttt{category}. I’ve chosen the latter method and also provided aliases for the custom \texttt{year} and \texttt{cast} fields:

\texttt{field-aliases={identifier=category,year=user1,cast=user2}, identical-sort-action={category}}
This ensures that books always come before films with the same title. An oddity is the film “Whisky Galore!” which is one character different from the book “Whisky Galore” but the default locale collator ignores punctuation so the two titles are considered identical by the collator (but not by \texttt{sort-suffix=non-unique}). If a letter comparison was used instead, they would no longer be considered identical, but in this case the film would still be placed after the book since the film title is longer.

Since I’ve set the \texttt{category} I can provide semantic formatting commands (as for \texttt{sample-bacteria.tex}):

\begin{verbatim}
\newcommand*{\bookfont}[1]{\textit{#1}}
\newcommand*{\filmfont}[1]{\textsf{\textit{#1}}}
\glssetcategoryattribute{book}{textformat}{bookfont}
\glssetcategoryattribute{book}{glossnamefont}{bookfont}
\glssetcategoryattribute{film}{textformat}{filmfont}
\glssetcategoryattribute{film}{glossnamefont}{filmfont}
\end{verbatim}

I’ve given films a slightly different format to make them easier to distinguish from books of the same name.

Both \texttt{books.bib} and \texttt{films.bib} had the custom \texttt{year} field, indicating the year of first publication or release, which I’ve assigned to the \texttt{user1} field. I can define post-name hooks for each category to append the year in brackets after the name is displayed in the glossary:

\begin{verbatim}
\newcommand*{\glsxtrpostnamebook}{%
  \ifgls\useglsfield{user1}{\glsentrylabel}{\space(published \glsentryfieldvalue)}%
  {}%
}\newcommand*{\glsxtrpostnamefilm}{%
  \ifgls\useglsfield{user1}{\glsentrylabel}{\space(released \glsentryfieldvalue)}%
  {}%
}\end{verbatim}

As with the post-description hook, if you have at least glossaries-extra v1.31, it’s better to use:

\glsdefpostname{\texttt{category}}

instead of:

\newcommand{\glsxtrpostname{\texttt{category}}}

as it can guard against accidental misspelling of the \texttt{glsxtrpostname} part of the command name.

I’ve assigned the \texttt{cast} field to the \texttt{user2} field, and since this field uses \LaTeX’s contributor markup I need to convert this to a form that’s easier to customize:
I’m not sorting by this field and it would look better in the document to list the forenames before the surname so I’ve also done:

\contributor-order={\forenames}

Since I have at least version 2.28 of datatool-base installed, the list will be formatted using \DTLformatlist. If I want an Oxford comma, I need to redefine \DTLlistformatoxford in the document:

\renewcommand*{\DTLlistformatoxford}{,}

If I want to change “&” to “and” I also need to redefine \DTLandname:

\renewcommand*{\DTLandname}{and}

If \DTLformatlist isn’t defined (datatool-base v2.27 or earlier), the list will look a little odd as it uses a comma separator between all elements of this list, including the final pair (so there’s no final & or “and”).

I’ve provided a post-description hook \glsxtrpostdesc\langle category \rangle to append the cast list:

\newcommand*{\glsxtrpostdescfilm}{\%}
\ifglsishasfield{user2}{\glscurrententrylabel}\%
\{\%
\glsxtrrestorepostpunc \% requires glossaries-extra v1.23+
\&featuring \glscurrentfieldvalue
\%
\}\
\%
\}

This uses \glsxtrrestorepostpunc to restore the post-description punctuation if it was suppressed with \glsxtrnopostpunc. This means that if I decide not to include the user2 field then the post-description punctuation will be revert back to being suppressed for entries containing \glsxtrnopostpunc in the description field.

I haven’t referenced any of the entries in the main body of the document, so I’ve used selection={all} to select all entries. This means that there are no number lists on the first document build (\LaTeX+\bib2gls+\LaTeX) but the next build would show locations for the books that have been referenced by the film entries. Since this looks a bit odd, I’ve added save-locations={false} to prevent \bib2gls from saving the locations.

I’ve used a style that shows letter group headings so I need to use the --group switch. The complete document code is listed below. The document build is:

pdflatex sample-media
\bib2gls --group sample-media
pdflatex sample-media

The four pages of the document are shown in figure 8.9.
\documentclass[11pt,a4paper]{report}

\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage[record, % using bib2gls
  nostyles, % don't load default styles
  postdot, % append a dot after descriptions
  stylemods={list}, % load glossary-list.sty and fix styles
  style=altlistgroup]{glossaries-extra}

\GlsXtrLoadResources[
  src=no-interpret-preamble,
  interpret-preamble=false
]

\GlsXtrLoadResources[
  src={interpret-preamble,books,films},
  field-aliases={identifier=category,year=user1,cast=user2},
  bibtex-contributor-fields={user2},
  contributor-order={forenames},
  identical-sort-action={category},
  save-locations=false,
  selection=all
]

% requires datatool-base.sty v2.28+:
\renewcommand*{\DTLlistformatoxford}{,}
\renewcommand*{\DTLandname}{and}

\newcommand*{\bookfont}[1]{\emph{#1}}
\newcommand*{\filmfont}[1]{\textsf{\em #1}}

\glssetcategoryattribute{book}{textformat}{bookfont}
\glssetcategoryattribute{book}{glossnamefont}{bookfont}
\glssetcategoryattribute{film}{textformat}{filmfont}
\glssetcategoryattribute{film}{glossnamefont}{filmfont}

\newcommand*{\glsxtrpostnamebook}{%
  \ifgls\hasfield{user1}\glscurrententrylabel
  \space(published \glscurrentfieldvalue)
  \%
  \}
}

\newcommand*{\glsxtrpostnamefilm}{%
  \ifgls\hasfield{user1}\glscurrententrylabel
  \%
  \}
This example uses the files people.bib, no-interpret-preamble.bib and interpret-preamble.bib. The aim here is to have a list of people ordered alphabetically by surname with a brief description, the same list ordered by date of birth and an index of all the people without their details but with a number list indicating where that person was mentioned in the document. The first two lists shouldn’t include aliases but the index should. Not all the entries defined in people.bib are included in the document. Those that aren’t either explicitly referenced or aliased are filtered by the selection criteria. I’ve used a style that shows letter group headings so I need to use the --group switch.

Since this is just an example document all the \gls commands only occur on page 1, which means that each number list is just “1”. A real document would have the references scattered about. The aliases haven’t actually been referenced anywhere in the document.

The born, died and othername fields will be ignored by default since they don’t correspond to recognised keys, so the keys either need to be defined or the fields need to be mapped to existing keys. In this case I’ve decided to map them to the user1, user2 and user3 fields using field-aliases:

\field-aliases={born=user1,\died=user2,othername=user3}

Although the aliases haven’t been referenced in the document, I’ve taken into account the possibility that they might later be added. To prevent them from showing in the first two lists I’ve filtered them out. This is easy to do since the aliases are all defined using @index whereas the remaining (non-aliased) entries are defined using @entry so match can be used to only select entries defined with @entry:

match={\entrytype=entry}
Glossary

**H**
The Big Sleep (published 1939)  
* a film based on the novel *The Big Sleep* featuring Humphrey Bogart and Lauren Bacall.

Whisky Galore! (published 1947)  
* a black-and-white film based on the novel *Whisky Galore!* featuring Robert Newton, Joan Greenwood, and John Harvey.

The Hobbit: The Desolation of Smaug (released 2013)  
* a film based on the novel *The Hobbit* featuring Ian McKellen, Martin Freeman, and Richard Armitage.

**B**
The Big Sleep (published 1939)  
* a film based on the novel *The Big Sleep* featuring Humphrey Bogart and Lauren Bacall.

Whisky Galore! (published 1947)  
* a black-and-white film based on the novel *Whisky Galore!* featuring Robert Newton, Roger Livesey, and Jane Wyman.

The Hobbit: The Desolation of Smaug (released 2013)  
* a film based on the novel *The Hobbit* featuring Ian McKellen, Martin Freeman, and Richard Armitage.

**M**
The Hobbit: The Desolation of Smaug (released 2013)  
* a film based on the novel *The Hobbit* featuring Ian McKellen, Martin Freeman, and Richard Armitage.

Where Eagles Dare (published 1967)  
* a film based on the novel *Where Eagles Dare* featuring Richard Burton, Clint Eastwood, and Mary Ure.

The Lord of the Rings: The Fellowship of the Ring (published 1954)  
* a film based on the novel *The Lord of the Rings* featuring Elijah Wood, Ian McKellen, and Orlando Bloom.

**T**
The Lord of the Rings: The Return of the King (released 2003)  
* a film based on the novel *The Lord of the Rings* featuring Elijah Wood, Viggo Mortensen, and Ian McKellen.

The Lord of the Rings: The Battle of Five Armies (released 2014)  
* a film based on the novel *The Hobbit* featuring Ian McKellen, Martin Freeman, and Richard Armitage.

**W**
The Hobbit: An Unexpected Journey (released 2012)  
* a film based on the novel *The Hobbit* featuring Martin Freeman, Ian McKellen, and Richard Armitage.

The Hobbit: The Battle of Five Armies (released 2014)  
* a film based on the novel *The Hobbit* featuring Ian McKellen, Martin Freeman, and Richard Armitage.

The Hobbit: The Desolation of Smaug (released 2013)  
* a film based on the novel *The Hobbit* featuring Ian McKellen, Martin Freeman, and Richard Armitage.

**I**
The Hobbit: The Battle of Five Armies (released 2014)  
* a film based on the novel *The Hobbit* featuring Ian McKellen, Martin Freeman, and Richard Armitage.
I’d like the first use of `\gls` to display the full name, except for the entry that has the `first` field set. The remaining entries only have `text` set to a shortened version of the name so they need to have the `name` field copied to the `first` field using `replicate-fields`:

```
replicate-fields={name={first}}
```

I’d like the first use to show the other name in parentheses where provided. The simplest way to achieve this is by defining the post-link hook `\glsxtrpostlink\langle category\rangle`. If the `category` field isn’t specified it will default to `general` (for entries defined with `@entry`), so I could just define `\glsxtrpostlinkgeneral` but to allow for the possibility of extending the document to incorporate other types of entries I decided to set the `category` to `people` through the use of the `category` option:

```
category={people}
```

This means that I now need to define a command called `\glsxtrpostlinkpeople` that will be used after instances of `\gls` etc where the entry has the `category` set to `people`. This first tests if that was the first use of the entry with `\glsxtrifwasfirstuse` and then tests if the `user3` field is set. If so, it does a space followed by that field’s value in parentheses. The entry’s label can be obtained from `\glslabel`:

```
\newcommand*{\glsxtrpostlinkpeople}{%  
  \ifglshasfield{user3}{\glslabel}%  
  {\space(\glscurrentfieldvalue)}%  
  {}%  
  {}% 
}
```

I’d also like to do something similar after the name when the entry is displayed in the glossary. This means defining the post-name hook `\glsxtrpostname\langle category\rangle`, in this case `\glsxtrpostnamepeople`. The entry’s label is referenced with `\glscurrententrylabel`:

```
\newcommand*{\glsxtrpostnamepeople}{%  
  \ifglshasfield{user3}{\glscurrententrylabel}%  
  {\space(\glscurrentfieldvalue)}%  
  {}%  
}
```

(A different command is used since `\gls` may occur in the description, which would interfere with the current entry label if they shared the same command to reference the label.)

The post-description hook can be used to append the birth and death dates. Although all the entries that have been selected from `people.bib` have a `died` field, I’ve added a check for the corresponding `user3` field in case new references are added for people who are still alive:
\newcommand*{\glsxtrpostdescpeople}{\%
  \ifglsfield{user1}{\glscurrententrylabel}
  {\% born
   \space(\glscurrentfieldvalue,--\,%,
   \ifglsfield{user2}{\glscurrententrylabel}
   {\% died
    \glscurrentfieldvalue
   }%
  }
}

The first list is quite straightforward and can be created with:

\GlsXtrLoadResources[
  src={people},
  match={entrytype=entry},
  category={people},
  replicate-fields={name={first}},
  field-aliases={born=user1,died=user2,othername=user3}
]

I have used the \texttt{sort} option and there’s no document language, so \texttt{bib2gls} will sort according to my locale. The custom commands \texttt{\sortname} and \texttt{\sortvonname} ensure that the entries are all sorted alphabetically according to the surnames.

The second list can easily be created by adding the \texttt{secondary} option:

\texttt{secondary={date:user1:bybirth}}

This sorts according to the \texttt{user1} field (which was originally the \texttt{birth} field). Note that different locales have different default date formats. There may also be a difference in the default date format depending on the Java locale provider. For example, if you switch from using the \texttt{JRE} to using the \texttt{CLDR} you may find a change in the default format. In case the format provided in the .\texttt{bib} file isn’t recognised, the required format can be set with:

\texttt{secondary-date-sort-format={d MMM YYYY G}}

I’ve changed the date group headings by redefining \texttt{\glsdategroup} and \texttt{\glsdategrouptitle}, which means that the grouping in the bybirth glossary will be in the form \texttt{⟨year⟩ ⟨era⟩}:

\newcommand{\glsdategroup}[7]{\#1\#4\#7}
\newcommand{\glsdategrouptitle}[7]{\number\#1 \#4}

I’ve also defined the bybirth glossary and supplied a title:
\newglossary*{bybirth}{People (Ordered by Birth)}

The first two glossaries have entries with fairly long names (especially those with the post-namehook), so the best style is the altlistgroup. The glossaries-extra-stylemods package patches this style to discourage page breaks occurring after group headings, so I’ve also used the stylemods option to automatically load that package. I’d like to use the bookindex style for the index, which is provided by glossary-bookindex, so I need:

\setglossstyle{list,bookindex}

This ensures that glossary-list and glossary-bookindex are loaded and patches the list styles.

The first two glossaries would look better with a terminating full stop, so I’ve used the postdot package option. (The bookindex style doesn’t use the description field and therefore doesn’t use the post-description hook.) The index glossary type can be defined with the index package option. I’ve set the default style to altlistgroup but this can locally be changed to bookindex when I display the index. The record option is needed to use bib2gls, so the glossaries-extra package is loaded with:

\usepackage[record,% using bib2gls
index,% create index glossary
postdot,% dot after descriptions
% load glossary-list.sty and glossary-bookindex.sty and patch:
  \setglossstyle{list,bookindex},
  style=altlistgroup]{glossaries-extra}

The index needs to include all the entries that have already been defined but also needs to include the aliased entries. This means that existing entries simply need their label copied to the index glossary but the other entries need to be defined so this requires setting the action option:

action={define or copy}

I would also like to have groups in the index (which the bookindex style supports) so I need to specify a field in which to save the group information using copy-action-group-field:

\setglosscfg{copy-action-group-field={indexgroup}}

I need to remember to redefine \glsxtrgroupfield to this value before displaying the index:

\renewcommand{\glsxtrgroupfield}{indexgroup}

The aliased entries won’t be selected by default since they haven’t been used in the document, so I need to change the selection criteria with selection:

selection={recorded and deps and see}
In the index, I’d like the surnames first. This can be done by redefining the custom commands used in the name fields. There’s a slight complication here. These commands aren’t defined on the first \LaTeX run as their definitions are written to the .glistex file by bib2gls, so I can’t use \renewcommand (although I could use \glsrenewcommand). Instead I’ve provided some custom commands:

\newcommand*{\swaptwo}{#2, #1}
\newcommand*{\swapthree}{#2 #3, #1}

Now I just need to make an assignment using \let:

\let\sortname\swaptwo
\let\sortart\swaptwo
\let\sortvonname\swapthree

This doesn’t perform any check to determine if the commands are already defined so there won’t be a problem on the first run.

The first two glossaries shouldn’t have number lists:

\printunsrtglossary[title={People (Alphabetical)},nonumberlist]
\printunsrtglossary[type={bybirth},target={false},nonumberlist]

I’d like to use hyperref but I have to switch off the hypertargets for the second glossary otherwise I’ll end up with duplicate targets. This is done with target={false}. All references using \gls etc will link to the first glossary.

I could also do this for the index but the cross-references in the aliased entries will link to the first glossary rather than the relevant entry in the index. The simplest way to fix this is to redefine \glolinkprefix to provide a different target:

\renewcommand*{\glolinkprefix}{idx:}

These redefinitions need to be done before the index. I’ve decided to use the starred \printunsrtglossary* to localise these changes, although that’s not needed for this document since the index comes right at the end:

\printunsrtglossary*
[type={index},style={bookindex}]
{%
    \let\sortname\swaptwo
\let\sortart\swaptwo
\let\sortvonname\swapthree
\renewcommand{\glsxtrgroupfield}{indexgroup}\
\renewcommand*{\glolinkprefix}{idx:}%
%
}

The complete document code is listed below. The document build is:
The four pages of the document are shown in figure 8.10.
\documentclass[12pt,a4paper]{report}

\usepackage[colorlinks]{hyperref}
\usepackage[record, % using bib2gls
  index, % create index glossary
  postdot, % dot after descriptions
  % load glossary-list.sty and glossary-bookindex.sty and patch:
  % stylemods={list,bookindex},
  % style=altlistgroup]{glossaries-extra}

\newglossary*{bybirth}{People (Ordered by Birth)}

\newcommand{\bibglsdategroup}[7]{#1#4#7}
\newcommand{\bibglsdategrouptitle}[7]{\number#1 \ #4}
\newcommand*{\swaptwo}[2]{#2, #1}
\newcommand*{\swapthree}[3]{#2 #3, #1}
\GlsXtrLoadResources[
  src=no-interpret-preamble,\n  interpret-preamble=false\n]
\GlsXtrLoadResources[
  src={interpret-preamble,people},
  match={entrytype=entry},
  category={people},
  replicate-fields={name={first}},
  field-aliases={born=user1,died=user2,othername=user3},
  secondary={date:user1:bybirth},
  secondary-date-sort-format={d MMM YYYY G}\n]
\GlsXtrLoadResources[
  src={people},
  type=index,
  category=people,
  action={define or copy},
  copy-action-group-field={indexgroup},
  selection={recorded and deps and see}\n]
8 Examples: sample-people.tex

\newcommand{*{\glsxtrpostlinkpeople}{\%
    \glsxtrifwasfirstuse
    {\% 
        \ifglshasfield{user3}{\glslabel}{\space(\glscurrentfieldvalue)}{\%
    }{\%
}{\%
}
\newcommand{*{\glsxtrpostnamepeople}{\%
    \ifglshasfield{user3}{\glscurrententrylabel}{\space(\glscurrentfieldvalue)}{\%
}{\%
}
\newcommand{*{\glsxtrpostdescpeople}{\%
    \ifglshasfield{user1}{\glscurrententrylabel}{\% born
        \space(\glscurrentfieldvalue,--\,\%
    \ifglshasfield{user2}{\glscurrententrylabel}{\% died
        \glscurrentfieldvalue
    }{\%
}{\%
}{\%
}
\begin{document}
\chapter{Sample}
\section{First Use}
\gls{caesar}, \gls{wellesley}, \gls{bonaparte}, \gls{vonrichthofen} and \gls{alexander}.

\section{Next Use}
\gls{caesar}, \gls{wellesley}, \gls{bonaparte}, \gls{vonrichthofen} and \gls{alexander}.

\printunsrtglossary[target=false,nonumberlist]
\printunsrtglossary[type=bybirth,nonumberlist]
This example uses the files `people.bib`, `books.bib`, `no-interpret-preamble.bib` and `interpret-preamble2.bib`. The aim is to reference the books in `books.bib` and have them listed by author. This means finding a way of assigning each book entry a `parent` field that contains the label identifying the relevant author in `people.bib`. I've used a style that shows letter group headings so I need to use the `--group` switch.

To recap, each author is defined in `people.bib` in the form:

```bibtex
@entry{dickens,
  name={\sortname{Charles}{Dickens}},
  text={Dickens},
  description={English writer and social critic},
  born={7~February 1812 AD},
  died={9~June 1870 AD},
  identifier={person}
}
```

and each book is defined in `books.bib` in the form:

```bibtex
@entry{bleakhouse,
  name={Bleak House},
  description={novel by Charles Dickens},
  identifier={book},
  author={\sortmediacreator{Charles}{Dickens}},
  year={1852}
}
```

There’s a field here (the custom `author` field) that contains the author’s name, and this can be aliased to the `parent` field with `field-aliases`:

```bibtex
field-aliases={author=parent}
```

but the author’s label in the `people.bib` file is just the lower case surname.
Chapter 1
Sample

1.1 First Use
Julius Caesar, Arthur Wellesley (1st Duke of Wellington), Napoleon Bonaparte, Manfred von Richthofen (The Red Baron) and Alexander III of Macedon (Alexander the Great).

1.2 Next Use
Caesar, Wellington, Bonaparte, von Richthofen and Alexander.

People (Alphabetical)

A
Alexander III of Macedon (Alexander the Great), Ancient Greek king of Macedon (20 July 356 BC – 10 June 323 BC).

B
Napoleon Bonaparte, French military and political leader (15 July 1769 AD – 5 May 1821 AD).

C
Gaius Julius Caesar, Roman politician and general (13 July 100 BC – 15 March 44 BC).

V
Manfred von Richthofen (The Red Baron), Prussian ace fighter pilot in the German Air Force during World War I (2 May 1892 AD – 21 April 1918 AD).

W
Arthur Wellesley (1st Duke of Wellington), Anglo-Irish soldier and statesman (1 May 1769 AD – 14 September 1852 AD).

People (Ordered by Birth)

357 BC
Alexander III of Macedon (Alexander the Great), Ancient Greek king of Macedon (20 July 356 BC – 10 June 323 BC).

100 BC
Gaius Julius Caesar, Roman politician and general (13 July 100 BC – 15 March 44 BC).

1769 AD
Napoleon Bonaparte, French military and political leader (15 July 1769 AD – 5 May 1821 AD).

1892 AD
Manfred von Richthofen (The Red Baron), Prussian ace fighter pilot in the German Air Force during World War I (2 May 1892 AD – 21 April 1918 AD).

1952 AD
Arthur Wellesley (1st Duke of Wellington), Anglo-Irish soldier and statesman (1 May 1769 AD – 14 September 1852 AD).

Index

A
Alexander III of Macedon (Alexander the Great), Ancient Greek king of Macedon (20 July 356 BC – 10 June 323 BC).

B
Napoleon Bonaparte, French military and political leader (15 July 1769 AD – 5 May 1821 AD).

C
Gaius Julius Caesar, Roman politician and general (13 July 100 BC – 15 March 44 BC).

V
Manfred von Richthofen (The Red Baron), Prussian ace fighter pilot in the German Air Force during World War I (2 May 1892 AD – 21 April 1918 AD).

W
Arthur Wellesley (1st Duke of Wellington), Anglo-Irish soldier and statesman (1 May 1769 AD – 14 September 1852 AD).

Figure 8.10: sample-people.pdf
Remember from chapter 2 that the interpreter will be used on the parent field if the value contains \ or { or } and interpret-label-fields={true}. This means that with this field alias and the interpreter on, bib2gls will attempt to interpret the field contents. So all that’s needed is to ensure that bib2gls is given a definition of \sortmediacreator that ignores the first argument and converts the second argument to lower case. This definition is available in interpret-preamble2.bib but, since this file uses \renewcommand rather than \providecommand, write-preamble={false} is required to prevent \LaTeX{} from picking up this definition.

As with the sample-people.tex example, I need to copy the name field to the first field if that field is missing using replicate-fields:

\texttt{replicate-fields={name={first}}}

and I also want to provide a semantic command to format the book title, so the field aliases also need to convert the custom identifier field to category:

\texttt{field-aliases={identifier=category,author=parent}}

so that the document can set the textformat and glossnamefont attributes:

\texttt{\newcommand*{\bookfont}[1]{\emph{#1}}
\glssetcategoryattribute{book}{textformat}{\bookfont}
\glssetcategoryattribute{book}{glossnamefont}{\bookfont}}

As with sample-media.tex, the terminating question mark at the end of some of the name fields can cause an awkward situation if \texttt{\gls} is used at the end of a sentence. This can be dealt with by getting bib2gls to make a note of the fields that end with sentence-terminating punctuation through the use of the check-end-punctuation option. In this example, the name, text and first fields are the same for all the books, so it’s sufficient just to check the name field:

\texttt{check-end-punctuation={name}}

With glossaries-extra v1.23+ it’s easy to hook into the post-link hook to check if nameendpunc exists:

\texttt{\renewcommand*{\glsxtrifcustomdiscardperiod}[2]{%
\GlsXtrIfFieldUndef{nameendpunc}{\glslabel}{#2}{#1}%
}}

This will now cause the full stops following:

\texttt{\gls{whydidn\textquotesingle ttheyaskevans}}.

and

\texttt{\gls{doandroidsdreamofelectricsheep}}.
to be discarded.

The complete document code is listed below. The document build is:

\texttt{pdflatex sample-authors}
\texttt{bib2gls --group sample-authors}
\texttt{pdflatex sample-authors}

The resulting document is shown in figure 8.11.

\begin{verbatim}
\documentclass[12pt,a4paper]{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record, % using bib2gls
  nostyles, % don't load default styles
  stylemods={bookindex}, % load glossary-bookindex and patch styles
  style=bookindex]{glossaries-extra}
\GlsXtrLoadResources[  
  src=no-interpret-preamble,  
  interpret-preamble=false
]
\GlsXtrLoadResources[  
  src={interpret-preamble2,people,books},  
  write-preamble=false,  
  interpret-label-fields,  
  field-aliases={identifier=category,author=parent},  
  check-end-punctuation={name},  
  replicate-fields={name={first}}
]
\newcommand*{\bookfont}[1]{\emph{#1}}
\glssetcategoryattribute{book}{textformat}{\bookfont}
\glssetcategoryattribute{book}{glossnamefont}{\bookfont}
\renewcommand*{\glsxtrifcustomdiscardperiod}[2]{%  
  \GlsXtrIfFieldUndef{nameendpunc}{\glslabel}{#2}{#1} %
}
\begin{document}
\section{Sample}
\gls{ataleoftwocities}. \gls{bleakhouse}. \gls{thebigsleep}.  
\gls{thelonggoodbye}. \gls{redharvest}.  
\gls{murderontheorientexpress}. \gls{whydidnttheyaskevans}.  
\gls{icecoldinalex}. \gls{thehobbit}. \gls{thelordoftherings}.  
\gls{thewonderfulwizardofoz}. \gls{whiskygalore}.
\end{document}
\end{verbatim}
8 Examples: sample-citations.tex

\gls{whereeaglesdare}. \gls{icestationzebra}. \gls{ubik}. \gls{doandroidsdreamofelectricsheep}. \gls{thetroublewithharry}. \gls{brightonrock}.

\printunsrtglossary[title={Author and Book List}]
\end{document}

sample-citations.tex

This example uses the \LaTeX\ file citations.bib to create a document that has both a bibliography created by \LaTeX\ and glossaries created by \texttt{bib2gls} listing the authors and the titles. There are no glossary reference commands, such as \gls, but \texttt{bib2gls} can be run with \texttt{--cite-as-record} to treat the \texttt{\citation} commands (written to the .aux file by \texttt{\cite}) as ignored records. Since \texttt{\cite} doesn’t record the page number, there are no associated locations.

The main glossary isn’t required, so I’ve used \texttt{nomain} to suppress its creation. I want to use both the altlist and indexgroup styles but none of the other styles, so I’ve used \texttt{nostyles} to prevent the automatic loading of the default style packages and \texttt{stylemods} to load the glossary-tree and glossary-list packages and patch the styles. A full stop is automatically placed after the descriptions with \texttt{postdot}.

\usepackage[record,\% using \texttt{bib2gls} nomain,\% don't define main glossary postdot,\% full stop after descriptions nostyles,\% don't load default styles \% load glossary-tree and glossary-list and patch styles: stylemods={tree,list} ]{glossaries-extra}

Next I need to create the glossaries for the list of authors and list of titles:

\newglossary*{contributors}{Authors}\newglossary*{titles}{Titles}

The simplest way of assigning the authors to the \texttt{contributors} glossary and the titles to the \texttt{titles} glossary is to use:

type={contributors}

in the resource set and provide a modified version of \texttt{\bibglsnewbibtexentry} that assigns \texttt{type} after the options:

\newcommand{\bibglsnewbibtexentry}[4]{\% \longnewglossaryentry*{#1}{name={#3},#2,type={titles}}{#4}\%}

494
1 Sample


**Author and Book List**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| B | Lyman Frank Baum  
_The Wonderful Wizard of Oz, 1_  
| C | Raymond Chandler  
_The Big Sleep, 1_  
_The Long Goodbye, 1_  
| L | Dame Agatha Mary Clarissa Christie  
_Murder on the Orient Express, 1_  
_Why Didn’t They Ask Evans?, 1_  
| H | Sammel Dashiel Hammett  
_Red Harvest, 1_  
| D | Philip K. Dick  
_Do Androids Dream of Electric Sheep?, 1_  
_Ubik, 1_  
| S | Charles Dickens  
_Bleak House, 1_  
_A Tale of Two Cities, 1_  
| M | Jack Trevor Story  
_The Trouble with Harry, 1_  
| G | Compton Mackenzie  
_Whisky Galore, 1_  
| T | Christopher Guy Landon  
_Ice Cold in Alex, 1_  
|   | Alistair MacLean  
_Ice Station Zebra, 1_  
_Where Eagles Dare, 1_  
|   | John Ronald Reuel Tolkien  
_The Hobbit, 1_  
_The Lord of the Rings, 1_  

Figure 8.11: sample-authors.pdf
The standard \LaTeX entry types need aliasing to \texttt{\@bibtexentry}:

\texttt{entry-type-aliases={\GlsXtrBibTeXEntryAliases}}

and the \texttt{title} field is aliased to \texttt{name}:

\texttt{field-aliases={title=name}}

(The other fields aren’t required for the glossary lists.) The \texttt{category} is set to the original entry type:

\texttt{category={same as original entry}}

So, for example, an entry that’s provided in the .bib file with \texttt{@article} has the \texttt{category} field set to \texttt{article}. (Compare this with \texttt{category={same as entry}} which would set the \texttt{category} to \texttt{bibtexentry}.) The spawned entries are all defined using \texttt{\@contributor} and aren’t aliased so both the entry type and the original entry type are \texttt{contributor}.

In order to list the titles according to category, I’ve use this as the sort field:

\texttt{sort-field={category}}

and setting the sort suffix to the \texttt{name} field sub-sorts the \texttt{\@bibtexentry} types according to the title (which was aliased to the \texttt{name}) and the \texttt{\@contributor} types according to the author:

\texttt{sort-suffix={name}}

Next the groups identified by the labels \texttt{article} and \texttt{book} are assigned titles.

\texttt{\glsxtrsetgrouptitle{article}{Articles}}
\texttt{\glsxtrsetgrouptitle{book}{Books}}

The \texttt{group} field is actually set to the associated letter by the default \texttt{sort} method. The desired labels are stored in the \texttt{category} field. Since the entries are sorted by category, then they are naturally in those sub-blocks, which means that the group titles can be set by locally redefining \texttt{\glsxtrgroupfield} to \texttt{category}:

\texttt{\%}
\texttt{%}
\texttt{\renewcommand{\glsxtrgroupfield}{category}%%}
\texttt{\renewcommand{\glstreenamefmt}[1]{\emph{#1}}}%%
\texttt{\renewcommand{\glstreegroupheaderfmt}[1]{\textbf{#1}}}%%
\texttt{\%}
\texttt{\%}

This again contradicts the advice given in section 1.2 as I’m sorting by the \texttt{group} label. (Technically it’s sorting by the \texttt{category} label but this is being used as the group.) In this case it’s not a problem as the labels closely match the titles and the sorting options ensure that the groups aren’t broken up.

There’s no \texttt{description} field set for these entries, but the post-description hook can still be used to append information. In this case, I’ve appended a cross-reference to the bibliography. Since the bibliography entry and the glossary term both have the same label, the citation can easily be obtained with \texttt{\cite{\glscurrententrylabel}}:
8 Examples: sample-citations.tex

\newcommand{\glsxtrpostdescarticle}{\cite{\glscurrententrylabel}}
\newcommand{\glsxtrpostdescbook}{\cite{\glscurrententrylabel}}

Note that this needs to be done for each \LaTeX entry type, but in this case the .bib file only contains @article and @book entries. (Similarly for the group titles above.)

The list of contributors can simply be displayed with:

\printunsrtglossary[type={contributors},style={altlist}]

This will only list the names as there’s no description, but again the post-description hook can be used, in this case for the contributor category. The hook iterates over the internal list provided by the \texttt{bibtexentry} field. This allows the titles to be listed as well:

\newcommand{\glsxtrpostdesccontributor}{%
  \glsxtrifhasfield{bibtexentry}{\glscurrententrylabel}{%
    \glsxtrfieldforlistloop{\glscurrententrylabel}{bibtexentry}{%}
    {\par No titles.}%
  }%
}

The handler macro displays the name of the associated \texttt{bibtexentry} term and the citation:

\newcommand{\contributorhandler}[1]{\par\glsentryname{#1} \cite{#1}}

The complete document code is listed below. The document build is:

\texttt{pdflatex sample-citations \bib2gls --cite-as-record sample-citations \bibtex sample-citations \pdflatex sample-citations \pdflatex sample-citations}

The resulting document is shown in figure 8.12.

\texttt{\documentclass[12pt,a4paper]{article}}

\texttt{\usepackage[record,% using \texttt{bib2gls}
nomain,% don't define main glossary
postdot,% full stop after descriptions
nostyles,% don't load default styles
% load glossary-tree and glossary-list and patch styles:
styloemods={tree,list}
]{glossaries-extra}}

\newglossary*{contributors}{Authors}
\newglossary*{titles}{Titles}
\newcommand{\bibglsnewbibtexentry}[4]{\longnewglossaryentry*[#1]{name={#3},#2,type={titles}}{#4}}
\GlsXtrLoadResources[src={citations},% data in citations.bib entry-type-aliases={\GlsXtrBibTeXEntryAliases}, field-aliases={
title=name },
type={contributors},
category={same as original entry},
sort-field={category},
sort-suffix={name}]
\glsxtrsetgrouptitle{article}{Articles}
\glsxtrsetgrouptitle{book}{Books}
\newcommand{\contributorhandler}[1]{\par\glsentryname{#1} \cite{#1}}
\newcommand{\glsxtrpostdesccontributor}{\glsxtrifhasfield{bibtexentry}{\glscurrententrylabel}{\glsxtrfieldforlistloop \glscurrententrylabel}{bibtexentry}{\contributorhandler}{\par No titles.}}
\newcommand{\glsxtrpostdescarticle}{\cite{\glscurrententrylabel}}
\newcommand{\glsxtrpostdescbook}{\cite{\glscurrententrylabel}}
\begin{document}
This is a sample document with some citations-\cite{macaw,parrot} and some more citations-\cite{duck2018,duck2016} and don't forget-\cite{ing,parrot2012} and lastly-\cite{quackalot}.
\printunsrtglossary[type=contributors,style=altlist]
\printunsrtglossary*[type=titles,style=indexgroup]{\renewcommand{\glsxtrgroupfield}{category}{\renewcommand{\glstreenamefmt}{\emph{#1}}}}
This example uses bigmathsymbols.bib, mathsrelations.bib, binaryoperators.bib, unaryoperators.bib and mathgreek.bib. The stix package is required for some of the commands used in bigmathsymbols.bib, so that must be loaded in the document.

I’m using the mcolalttree style for this document, which means that the glossary–mcols package is required and the styles need patching, which can be done with the stylemods package option:

\usepackage[record,\% using bib2gls
nostyles,\% don’t load default styles
postdot,\% append a dot after descriptions

8 Examples: sample-msymbols.tex

\texttt{stylemods=\{mcols\},% load glossary-mcols.sty and patch style=\{mcolalttree\}}\{glossaries-extra\}

I’m not referencing any of the entries in the document as I’m just generating a complete list of all the defined symbols. This means I need to tell bib2gls to select all entries and don’t bother saving the \texttt{location} field:

\texttt{save-locations={false},}
\texttt{selection={all}}

Since I’m using a style that’s based on alttree I need to find the widest \texttt{name}, which can be done with \texttt{set-widest}.

The simplest way of dividing the glossary into logical blocks is to sort according to the category, but first I need to use \texttt{field-aliases} to convert the custom \texttt{identifier} field to \texttt{category}:

\texttt{field-aliases={identifier=category}}

and sort by the \texttt{category} field:

\texttt{sort-field={category}}

Since this will cause identical sort values, I need to provide a a way of ordering these identical values. Here I’ve decided to fallback on the \texttt{description} field:

\texttt{identical-sort-action={description}}

This means that entries will be ordered by \texttt{category} and then \texttt{description}, which naturally creates blocks of symbol types in the glossary. This only uses a simple case-sensitive string comparison which is fine for English, but for another language it would be better to use \texttt{sort-suffix} as in the \texttt{sample-textsymbols.tex} file.

Remember that I want a small vertical gap between each logical block. These need the \texttt{group} field which, with the default locale sort, is obtained from the first letter of the sort value. In this case the sort value is obtained from the \texttt{category} field, and as each category happens to start with a different letter, this means I get the desired effect. However, in the event that I add more entries with a new category that happens to start with the same letter as an existing category, it’s better to provide a more future-proof method, so I’ve set the \texttt{group} field to fetch its value from the \texttt{category} field:

\texttt{replicate-fields={category=group}}

(Since the \texttt{field-aliases} option is always performed before \texttt{replicate-fields}, the \texttt{category} field will already have been set and is available for replicating.)

This means that I’m essentially sorting by the \texttt{group} labels, which this manual has warned against doing. In this case, it’s an acceptable break from that rule as I’ve used options that ensure the groups aren’t broken up during sorting and I’m not concerned with the group titles. A method such as that used in \texttt{sample-textsymbols2.tex} would end up with titled blocks, which I don’t want here. By using resource options such as \texttt{field-aliases} and \texttt{replicate-fields} I can avoid the warning that’s triggered with the default \texttt{--warn-non-bib-fields}.

The complete document code is listed below. The document build is:
The resulting document is shown in figure 8.13.
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{stix}
\usepackage[record,% using bib2gls
nostyles,% don't load default styles
postdot,% append a dot after descriptions
stylemods={mcols},% load glossary-mcols.sty and patch
style=mcolalttree]{glossaries-extra}
\GlsXtrLoadResources[
  src={bigmathsymbols,mathgreek,
      mathsrelations,binaryoperators,unaryoperators},
  sort-field={category},
  identical-sort-action={description},
  field-aliases={identifier=category},
  replicate-fields={category=group},
  set-widest,
  save-locations=false,
  selection=all
]
\begin{document}
\printunsrtglossaries
\end{document}

\section*{sample-maths.tex}

This example uses bigmathsymbols.bib and mathobjects.bib. It has a fairly similar preamble to sample-msymbols.tex, but no-interpret-preamble.bib and interpret-preamble.bib are now needed to provide the \sortart command:

\GlsXtrLoadResources[
  src={no-interpret-preamble},
  interpret-preamble={false}
]

There's also an extra custom field to alias:

field-aliases={identifier=category,format=user1}
8 Examples: sample-maths.tex

Glossary

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>÷</td>
<td>division</td>
</tr>
<tr>
<td>×</td>
<td>multiplication</td>
</tr>
<tr>
<td>−</td>
<td>subtraction</td>
</tr>
<tr>
<td>∮ ∮</td>
<td>contour integral</td>
</tr>
<tr>
<td>∬ ∬</td>
<td>double integral</td>
</tr>
<tr>
<td>∫ ∫</td>
<td>integral</td>
</tr>
<tr>
<td>∮ ∮</td>
<td>surface integral</td>
</tr>
<tr>
<td>∬ ∬ ∬</td>
<td>triple integral</td>
</tr>
<tr>
<td>∮ ∮ ∮</td>
<td>volume integral</td>
</tr>
<tr>
<td>α</td>
<td>alpha</td>
</tr>
<tr>
<td>β</td>
<td>beta</td>
</tr>
<tr>
<td>χ</td>
<td>chi</td>
</tr>
<tr>
<td>δ</td>
<td>delta</td>
</tr>
<tr>
<td>ε</td>
<td>epsilon</td>
</tr>
<tr>
<td>ϵ</td>
<td>epsilon (variant)</td>
</tr>
<tr>
<td>η</td>
<td>eta</td>
</tr>
<tr>
<td>γ</td>
<td>gamma</td>
</tr>
<tr>
<td>ι</td>
<td>iota</td>
</tr>
<tr>
<td>κ</td>
<td>kappa</td>
</tr>
<tr>
<td>λ</td>
<td>lambda</td>
</tr>
<tr>
<td>μ</td>
<td>mu</td>
</tr>
<tr>
<td>ν</td>
<td>nu</td>
</tr>
<tr>
<td>ω</td>
<td>omega</td>
</tr>
<tr>
<td>θ</td>
<td>theta</td>
</tr>
<tr>
<td>ϑ</td>
<td>theta (variant)</td>
</tr>
<tr>
<td>υ</td>
<td>upsilon</td>
</tr>
<tr>
<td>υ</td>
<td>upsilon</td>
</tr>
<tr>
<td>υ</td>
<td>upsilon</td>
</tr>
<tr>
<td>χ</td>
<td>zeta</td>
</tr>
<tr>
<td>≈</td>
<td>approximately</td>
</tr>
<tr>
<td>=</td>
<td>equals</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>≥</td>
<td>greater than or equal to</td>
</tr>
<tr>
<td>∈</td>
<td>in</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>≤</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>≫</td>
<td>much greater than</td>
</tr>
<tr>
<td>≪</td>
<td>much less than</td>
</tr>
<tr>
<td>≠</td>
<td>not equals</td>
</tr>
<tr>
<td>∉</td>
<td>not in</td>
</tr>
<tr>
<td>!</td>
<td>factorial</td>
</tr>
<tr>
<td>∀</td>
<td>for all</td>
</tr>
<tr>
<td>−</td>
<td>minus</td>
</tr>
<tr>
<td>+</td>
<td>plus</td>
</tr>
</tbody>
</table>

Figure 8.13: sample-msymbols.pdf
I’ve aliased \texttt{format} to \texttt{user1} since \texttt{\glsxtrfmt} defaults to that field. If I decided to use a different field I also need to remember to redefine \texttt{\GlsXtrFmtField} to match.

As with \texttt{sample-msymbols.tex} I’m sorting by the \texttt{category} label and this value is copied to the \texttt{group} field, but again I don’t have a hierarchical glossary as the logical blocks don’t have titles.

In this document I only want to select entries that have been indexed, so I’ve omitted the \texttt{selection} option I used in the \texttt{sample-msymbols.tex} example, however I still don’t want any number lists so I still have \texttt{save-locations=\{false\}}.

I want \texttt{\glsxtrfmt} to index the term (which it doesn’t by default) so that means I need to redefine \texttt{\GlsXtrFmtDefaultOptions} to prevent it from using \texttt{noindex}:

\begin{verbatim}
\renewcommand{\GlsXtrFmtDefaultOptions}{}
\end{verbatim}

I’ve provided some convenient wrapper commands that use \texttt{\glsxtrfmt*} or the non-linking \texttt{\glsxtrentryfmt} that are in the form:

\begin{verbatim}
\newcommand{\set}[2][]{\glsxtrfmt*[#1]{set}{#2}}
\newcommand{\nlset}[1]{\glsxtrentryfmt{set}{#1}}
\end{verbatim}

The use of the starred form allows:

\begin{verbatim}
[\set{A} = \gls{bigcup}\_\{i=1\}^n \set{B}\_i
\end{verbatim}

which produces:

\[ A = \bigcup_{i=1}^n B_i \]

Note the difference if the optional arguments aren’t used:

\begin{verbatim}
[\set{A} = \gls{bigcup}\_\{i=1\}^n \set{B}\_i
\end{verbatim}

This produces:

\[ A = \bigcup_{i=1}^n B_i \]

Be careful with the set cardinality example. You might be tempted to nest \texttt{\set} within the argument of \texttt{\setcard} but this results in nested hyperlinks. These are unpredictable and there’s no consistent handling of them between different PDF viewers. It can also be confusing to the reader. If \(|B_1 \cup B_2|\) shows up as what appears to be a single hyperlink, where would the reader expect the target? This is the reason for providing the non-linking commands like \texttt{\nlset} and \texttt{\nlsetcard}.

The complete document code is listed below. The document build is:

\begin{verbatim}
pdflatex sample-maths
bib2gls sample-maths
pdflatex sample-maths
\end{verbatim}

The resulting document is shown in figure 8.14.
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{amssymb}
\usepackage[record, % using bib2gis
  nostyles,% don’t load default styles
  postdot, % append a dot after descriptions
  stylemods={mcols},% load glossary-mcols.sty and patch
  style=mcolalttree]{glossaries-extra}
\GlsXtrLoadResources[
  src={no-interpret-preamble},
  interpret-preamble=false
]
\GlsXtrLoadResources[
  src={interpret-preamble,bigmathsymbols,mathsobjects},
  sort-field={category},
  identical-sort-action={description},
  field-aliases={identifier=category,format=user1},
  replicate-fields={category=group},
  set-widest,
  save-locations=false
]
\renewcommand{\GlsXtrFmtDefaultOptions}{}% requires glossaries-extra.sty v1.23+
\newcommand{\set}[2]{\glsxtrfmt*{set}{#1}{#2}}
\newcommand{\nlset}[1]{\glsxtrentryfmt{set}{#1}}
\newcommand{\setcontents}[2]{\glsxtrfmt*{setcontents}{#1}{#2}}
\newcommand{\setmembership}[2]{\glsxtrfmt*{setmembership}{{#1}{#2}}}
\newcommand{\nlsetcard}[1]{\glsxtrentryfmt{setcard}{#1}}
\newcommand{\transpose}[2]{\glsxtrfmt*{transpose}{#1}{#2}}
\newcommand{\nlttranspose}[2]{\glsxtrentryfmt{transpose}{#1}{#2}}
\newcommand{\inv}[2]{\glsxtrfmt*{inverse}{#1}{#2}}
\newcommand{\nlinv}[1]{\glsxtrentryfmt{inverse}{#1}}
\newcommand{\Vtr}[2]{\glsxtrfmt*{vector}{#1}{#2}}
\newcommand{\nlVtr}[1]{\glsxtrentryfmt{vector}{#1}}
\newcommand{\Mtx}[2]{\glsxtrfmt*{matrix}{#1}{#2}}
\newcommand{\nlMtx}[1]{\glsxtrentryfmt{matrix}{#1}}
\begin{document}
\section{Sets}

The universal set ($\gls{universalset}$) contains everything.

The empty set ($\gls{emptyset}$) contains nothing.

Some assignments:
\[
\begin{align*}
\set{B}[._1] &= \setcontents{1, 3, 5, 7}, \\
\set{B}[._2] &= \setcontents{2, 4, 6, 8}, \\
\set{B}[._3] &= \setcontents{9, 10}
\end{align*}
\]

Define:
\[
\begin{align*}
\set{A} &= \gls{bigcup}[_{i=1}^3 \set{B}[._i] \\
&= \setcontents{1, \ldots, 10}
\end{align*}
\]

The cardinality of a set $\gls{set}$ is denoted $\gls{setcard}$
and is the number of elements in the set.
\[
\begin{align*}
\setcard{\set{B}[._1]} &= 4, \\
\setcard{\set{B}[._2]} &= 4, \\
\setcard{\set{B}[._3]} &= 2, \\
\setcard{\set{B}[._1] \cup \set{B}[._2]} &= 8, \\
\setcard{\gls{emptyset}} &= 0
\end{align*}
\]

\section{Spaces}

A number space (denoted $\gls{numberspace}$) is characterised
by a set of entities with a set of axioms. For example:
\begin{align*}
\gls{naturalnumbers} &= \setmembership{x}{x \text{ is positive integer}} \\
\gls{integernumbers} &= \setmembership{x}{x \text{ is an integer}} \\
\gls{realnumbers} &= \setmembership{x}{x \text{ is a real number}}
\end{align*}

\section{Vectors and Matrices}

A matrix (denoted $\gls{matrix}$) is a rectangular array of values.
A vector (denoted $\gls{vector}$) is a column or row of values (that
is a one-dimensional matrix).
\[
\begin{align*}
\gls{identitymatrix}\Vtr{x} &= \Vtr{x}, \\
\Mtx{A}\inv{\Mtx{A}} &= \gls{identitymatrix}, \\
\inv{\Vtr{x}}\gls{1vec} &= \gls{sum}[._i] x_i
\end{align*}
\]

\printunsrtglossaries
8 Examples: sample-textsymbols.tex

\end{document}

sample-textsymbols.tex

This example uses miscsymbols.bib. This requires both marvosym and (with the weather option) ifsym. Unfortunately both define the commands \Sun and \Lightning, so these commands need to be undefined after the first package is loaded and before the second. Since I want the definitions provide by ifsym I have to first load marvosym, then undefine the conflicting commands and then load ifsym:

\usepackage{etoolbox}
\usepackage{marvosym}
\undef\Sun
\undef\Lightning
\usepackage[weather]{ifsym}

The etoolbox package is also loaded as it provides \undef. (An alternative is to modify the miscsymbols.bib file so that it uses ifsym’s more generic \textweathersymbol command and omit the weather option when loading the package, but the method used here demonstrates how to deal with such conflicts.)

The custom entry type @icon must be aliased for the entries to be recognised:

entry-type-aliases={icon=symbol}

Since none of the entries have a name or description field, the custom fields icon and icondescription need to be aliased to them. The document uses the alttreegroup style where the groups are obtained from the category, which again I obtain from the custom identifier field using:

field-aliases={
  identifier=category,
  icon=name,
  icondescription=description},
replicate-fields={category=group}

The group field is just a label and an appropriate title needs to be supplied for each group label:

\glsxtrsetgrouptitle{information}{Information}
\glsxtrsetgrouptitle{mediacontrol}{Media Controls}
\glsxtrsetgrouptitle{weather}{Weather Symbols}

This also requires sorting first by category and then fallback on another field. The most appropriate here is the description field, but instead of using identical-sort-action, I’m using sort-suffix, which works better with the default locale sort when the fallback field consists of words or phrases.
1 Sets

The universal set (\(U\)) contains everything. The empty set (\(\emptyset\)) contains nothing. Some assignments:

\[
B_1 = \{1, 3, 5, 7\}, \quad B_2 = \{2, 4, 6, 8\}, \quad B_3 = \{9, 10\}
\]

Define:

\[
A = \bigcup_{i=1}^{3} B_i = \{1, \ldots, 10\}
\]

The cardinality of a set \(S\) is denoted \(|S|\) and is the number of elements in the set.

\[
|B_1| = 4, \quad |B_2| = 4, \quad |B_3| = 2, \quad |B_1 \cup B_2| = 8, \quad |\emptyset| = 0
\]

2 Spaces

A number space (denoted \(S\)) is characterised by a set of entities with a set of axioms. For example:

\[
N = \{x : x \text{ is a positive integer}\}
\]

\[
Z = \{x : x \text{ is an integer}\}
\]

\[
R = \{x : x \text{ is a real number}\}
\]

3 Vectors and Matrices

A matrix (denoted \(M\)) is a rectangular array of values. A vector (denoted \(v\)) is a column or row of values (that is a one-dimensional matrix).

\[
Ix = x, \quad AA^{-1} = I, \quad x^{-1} = \sum_{i=1}^{n} z_i
\]

Glossary

- \(I\): the identity matrix.
- \(Z\): the set of integers.
- \(M^{-1}\): the inverse of \(M\).
- \(N\): the set of natural numbers.
- \(M\): a matrix.
- \(R\): the set of real numbers.
- \(v\): a vector.
- \(\mathbf{1}\): the vector of 1s.
- \(|S|\): the cardinality of \(S\).
- \(\emptyset\): the empty set.
- \(\sum\sum\): \(n\)-ary summation.
- \(\bigcup\bigcup\): \(n\)-ary union.
- \{\ldots\}: set contents.
- \(\{x : \ldots\}\): set membership.
- \(S\): a number space.
- \(U\): the universal set.

Figure 8.14: sample-maths.pdf
8 Examples: sample-textsymbols.tex

\begin{verbatim}
sort-field={category},
sort-suffix={description},
sort-suffix-marker={|

Since I’m using one of the alttree styles, I need to set the widest name:

set-widest

In this case, bib2gls won’t be able to determine the widest name since it doesn’t recognise any of the commands, so it will have to use the fallback command, which will use one of the commands provided by the glossaries-extra-stylemods package.

This is actually not the best method as bib2gls can’t see the group titles as they’re in the document, so it’s only able to sort by the label. While this might work for English, it can become a problem for other languages that use extended Latin or non-Latin characters in their alphabet. A much better method is to treat this as a hierarchical glossary with topic titles as the top-level entries. This is covered in the next example sample-textsymbols2.tex.

The complete document code is listed below. The document build is:

pdflatex sample-textsymbols
bib2gls sample-textsymbols
pdflatex sample-textsymbols

The resulting document is shown in figure 8.15.
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{etoolbox}
\usepackage{marvosym}
% package conflict, need to undefine conflicting commands
\undef\Sun
\undef\Lightning
\usepackage[weather]{ifsym}
\usepackage[record,% using bib2gls
nostyles,% don't load default styles
postdot,% append a dot after descriptions
stylemods={tree},% load glossary-tree.sty and patch
style=alttreegroup]{glossaries-extra}
\GlsXtrLoadResources[
  src={miscsymbols},
  % make @icon behave like @symbol:
  entry-type-aliases={icon=symbol},
  field-aliases={
\end{verbatim}

508
sample-textsymbols2.tex

This example is a better approach than the sample-textsymbols.tex example above. As with the previous example, this requires both marvosym and ifsym so the same patch is applied to avoid conflict.

As before, the custom entry type @icon must be aliased for the entries to be recognised:

```latex
entry-type-aliases={icon=symbol}
```

The topics.bib file contains terms with labels that match the custom identifier fields used in the micsymbols.bib file. So both files are loaded and the identifier field is now aliased to parent. These parent entries represent the topics and unlike the previous example it’s now possible to sort by the topic title (obtained from the name field) instead of by the label.

```latex
src={topics,miscsymbols},
field-aliases={
  identifier=parent,
  icon=name,
  icondescription=description},
```

There’s no sort-field option in this example. The default sort field is used. Since it’s not set for any of the entries, the fallback value will be used. In the case of the topic titles (@index and @indexplural), I want to sort by the name, which is the default fallback if the sort field is missing for the index entry types.

The default fallback for the sort field for @symbol entries is the label, but in this case I want to use the description field:
Glossary

Information
- bicycle route.
- café.
- football stadium.
- Gents.
- information centre.
- Ladies.
- recycling centre.
- wheelchair access provided.

Media Controls
- back to start of track.
- next track.
- play.
- rewind.

Weather Symbols
- cloudy.
- drizzle.
- foggy.
- hail.
- misty.
- overcast.
- rain.
- snow.
- sunny.
- thunderstorm.
The best styles for this kind of glossary are the topic styles provided by glossary-topic. This package was only added to glossaries-extra v1.40, so you need to make sure you have at least that version installed.

In this case I’ve decided to use the topic style. I can use it with or without the set-widest option. As with the previous example, bib2gls won’t be able to determine the widest name since it doesn’t recognise any of the commands contained in the name fields, so it will have to use the fallback method, which will use one of the commands provided by the glossaries-extra-stylemods package. The tree option is needed to enable the appropriate commands:

\usepackage[record,
nostyles,
postdot,
stylenods={tree,topic},
style={topic}]{glossaries-extra}

The complete document code is listed below. The document build is:

pdflatex sample-textsymbols2
bib2gls --group sample-textsymbols2
pdflatex sample-textsymbols2

The resulting document is shown in figure 8.16.
sample-markuplanguages.tex

This example uses markuplanguages.bib. Since the file includes abbreviations, any commands that must be used before abbreviations are defined need to go before \GlsXtrLoadResources. This includes the abbreviation style, which I’ve set to long-short-desc:

\setabbreviationstyle[markuplanguage]{long-short-desc}

This style sets the name field using \glsxtrlongshortdescname, which defaults to the long form followed by the short form in parentheses. I decided to switch this round so that the short form is shown first, which conveniently matches the default abbreviation-sort-fallback.

\renewcommand*{\glsxtrlongshortdescname}{%
  \protect\glsabbrvfont{\the\glsshorttok}\space
  \glsxtrparen{\glslongfont{\the\glslongtok}}%}

(The long form is still shown before the short form on the first use of \gls in the document. The switch in the above code only affects how the term is displayed in the glossary.)

This redefinition must be done before the abbreviations are defined as it’s expanded when the name field is set. (Note the need to protect commands that shouldn’t be expanded.) If I decide not to change the name format in this way, I would then need to use abbreviation-sort-fallback={long}.

I also decided to make use of the custom command \abbrvtag that marks up the letters in the long field used to obtain the abbreviation. As with the abbreviation style, this must be done before the abbreviations are defined:

\GlsXtrEnableInitialTagging{markuplanguage}{\abbrvtag}

If you accidentally place it after \GlsXtrLoadResources, you’ll encounter an error on the second \LaTeX run (but not the first). This is because \GlsXtrEnableInitialTagging requires that the supplied command (\abbrvtag in this case) be undefined. On the first \LaTeX run,

field-aliases={
  identifier=parent,
  icon=name,
  icondescription=description
},
symbol-sort-fallback={description},
set-widest,
selection=all
]

\begin{document}
\printunsrtglossaries
\end{document}
Glossary

Information

- bicycle route.
- café.
- football stadium.
- Gents.
- information centre.
- Ladies.
- recycling centre.
- wheelchair access provided.

Media controls

- back to start of track.
- next track.
- play.
- rewind.

Weather symbols

- cloudy.
- drizzle.
- foggy.
- hail.
- misty.
- overcast.
- rain.
- snow.
- sunny.
- thunderstorm.
it’s undefined, but on the second it picks up the \texttt{@preamble} definition, which is now in the resource file.

The tagging format is governed by \texttt{\glsxtrtagfont} which underlines its argument by default. I’ve redefined it to also convert the letter to upper case:

\begin{verbatim}
\renewcommand*{\glsxtrtagfont}[1]{\underline{\glsuppercase{#1}}}
\end{verbatim}

Note that in the \texttt{mathml} case, the first tag consists of more than one letter:

\begin{verbatim}
long={\abbrvtag{m\NoCaseChange{NoCaseChange{ath}}}ematical }#markuplang
\end{verbatim}

Here \texttt{\NoCaseChange} prevents \texttt{\glsuppercase} from applying the case change.

The default selection criteria includes entries that have been indexed and any cross-references. Some of the description fields include \texttt{\glsxtrshort}, which \texttt{bib2gls} picks up and the referenced entry is included in the dependency list. However, I don’t want any indexing performed by commands occurring in the glossary. This can be dealt with in one of two ways: either switch the format to \texttt{glsignore} or suppress the indexing by changing the default options with \texttt{\GlsXtrSetDefaultGlsOpts}. In this case I decided to turn the records into ignored records:

\begin{verbatim}
\GlsXtrSetDefaultNumberFormat{\texttt{glsignore}}
\end{verbatim}

This means that some of the entries won’t have location lists, so I’ve defined a post-description hook that inserts a full stop after the description if there’s no location otherwise it inserts a comma:

\begin{verbatim}
\newcommand{\glsxtrpostdescmarkuplanguage}{%
    \glsxtrifhasfield{location}{\glscurrententrylabel}%
    {,}%
    {.}%
}
\end{verbatim}

I’ve used \texttt{loc-suffix} to append a full stop after the location lists. This doesn’t affect the entries that haven’t been indexed.

I decided to convert the first letter of the name field to upper case. Since the name is implicitly set for abbreviations based on the style, I’ve decided to implement this through the \texttt{glossname} attribute rather than using \texttt{name-case-change}:

\begin{verbatim}
\glssetcategoryattribute{markuplanguage}{\texttt{glossname}}{firstuc}
\end{verbatim}

If this line causes an error when the glossary is displayed that goes away if it’s commented out, make sure you have at least version 2.06 of \texttt{mfirstuc}. For most of the entries, this doesn’t make a difference as they already start with a capital. It’s only the markdown entry that’s actually affected.

The description case change is dealt with by \texttt{bib2gls} instead:

\begin{verbatim}
description-case-change={\texttt{firstuc}}
\end{verbatim}
This works better than the glossdesc attribute as bib2gls can convert commands like \glstext into \Glstext which \makefirstuc can’t do. (Although in this particular example, there’s no difference as both instances of \glstext already produce upper case text.)

The complete document code is listed below. The document build is:

pdflatex sample-markuplanguages
bib2gls --group sample-markuplanguages
pdflatex sample-markuplanguages

The resulting document is shown in figure 8.17.

\documentclass[fontsize=12pt]{scrartcl}
\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage[record,% use bib2gls
nostyles,% don’t load default styles
% load glossary-tree.sty and patch styles:
stylesmods={tree},
style=treegroup]{glossaries-extra}
% abbreviation style must be set before \GlsXtrLoadResources
\GlsXtrEnableInitialTagging{markuplanguage}{long-short-desc}
\renewcommand*{\glsxtrlongshortdescname}{% 
  \protect\protect\glsabbrvfont{the}\glsshorttok space 
  \glsxtrparen{\glslongfont{the}\glslongtok}}%
\GlsXtrLoadResources[
  src=markuplanguages,% data in markuplanguages.bib
  loc=suffix,
  category=markuplanguage,
  description-case-change=firstuc
]
\newcommand{\glsxtrpostdescmarkuplanguage}{% 
  \glsxtrifhasfield{location}{\glscurrententrylabel}% 
  {,}% 
  {.}%
}
\glssetcategoryattribute{markuplanguage}{glossname}{firstuc}
\renewcommand*{\glsxtrtagfont}{\underline{\glsuppercase{#1}}}
\begin{document}
\section{First Use}
\gls{LaTeX}, \gls{markdown}, \gls{xhtml}, \gls{mathml}, \gls{svg}.
\section{Next Use}
\gls{LaTeX}, \gls{markdown}, \gls{xhtml}, \gls{mathml}, \gls{svg}.
\GlsXtrSetDefaultNumberFormat{glsignore}
\printunsrtglossary
\end{document}

\texttt{sample-usergroups.tex}

This example uses \texttt{usergroups.bib}. This requires \texttt{XeLaTeX} or \texttt{LuaLaTeX} as the \texttt{.bib} file includes non-ASCII labels. The entries include fields in different languages, the main one being English. If an entry has a non-English \texttt{name} or \texttt{long} field, it also includes the custom field \texttt{translation} that provides an (approximate) translation. If this field is present, the language is given by the first element of the custom \texttt{language} field.

In this case, I’m providing keys for the custom \texttt{language} and \texttt{translation} fields, and, for a bit of variety from the other examples, I’m ignoring the custom \texttt{identifier} field. The custom keys are provided with \texttt{\glsaddstoragekey}:

\begin{verbatim}
\glsaddstoragekey{language}{}{\glsentrylanguage}
\glsaddstoragekey{translation}{}{\glsentrytranslation}
\end{verbatim}

The \texttt{.bib} file includes abbreviations. Remember that the abbreviation style must be set before the resource file is loaded:

\begin{verbatim}
\setabbreviationstyle[tug]{long-short-user}
\end{verbatim}

For this example, I’m explicitly setting the \texttt{category} field to \texttt{tug}:

\begin{verbatim}
category={tug}
\end{verbatim}

Some of the fields end with a full stop. This isn’t a problem with the \texttt{long} field as the first use follows the long form with the short form in parentheses, but it will be a problem on subsequent use if the \texttt{short} field ends with a full stop. This means I need to check for end-of-sentence punctuation for the \texttt{short} field. It’s also a good idea to do this for the \texttt{name} field for the non-abbreviations.

\begin{verbatim}
check-end-punctuation={name,short}
\end{verbatim}
1 First Use
\LaTeX, markdown, extensible hypertext markup language (XHTML), mathematical markup language (MathML), scalable vector graphics (SVG).

2 Next Use
\LaTeX, markdown, XHTML, MathML, SVG.

Glossary:

H

HTML (HyperText Markup Language) The standard markup language for creating web pages.

L

\LaTeX A format of \LaTeX designed to separate content from style, 1

M

Markdown A lightweight markup language with plain text formatting syntax, 1

MathML (Mathematical Markup Language) Markup language for describing mathematical notation, 1

S

SVG (Scalable Vector Graphics) XML-based vector image format, 1

T

\TeX A format for describing complex type and page layout often used for mathematics, technical, and academic publications.

X

XHTML (eXtensible HyperText Markup Language) XML version of HTML, 1

XML (eXtensible Markup Language) A markup language that defines a set of rules for encoding documents.
It’s now possible to discard a full stop that follows \gls:

\renewcommand*{\glsxtrifcustomdiscardperiod}[2]{{\%  
  if\gls\hasshort{\glslabel}\%  
    \glsxtrifwasfirstuse\%  
    \GlsXtrIfFieldUndef{shortendpunc}{\glslabel}{#2}{#1}\%  
  }\%  
}  
\GlsXtrIfFieldUndef{nameendpunc}{\glslabel}{#2}{#1}\%
}\%)

This first tests if the entry that’s just been referenced has a short field. If it has, then the next test is to check if that was the first use for that entry. If it was, nothing is done. If it wasn’t, then \GlsXtrIfFieldUndef is used to determine if shortendpunc has been set. If it has been set then the period discard function is performed. If the entry doesn’t have a short field, then the nameendpunc field needs checking instead.

Since the document requires \TeX{} or \LaTeX{} and has some non-ASCII characters, it needs \fontspec{} and an appropriate font. In this case I’ve chosen “Linux Libertine O”. If you don’t have it installed, you’ll need to change it.

\usepackage{fontspec}
\setmainfont{Linux Libertine O}

Since it’s a multilingual document I also need \polyglossia{} with the main language set to \texttt{english}:

\usepackage{polyglossia}
\setmainlanguage[variant=uk]{english}

Now comes the difficult bit. The document needs to determine what other languages need to be loaded. The \tracklang{} package provides a convenient interface when dealing with language tags. This is automatically loaded by \glossaries{} but I’ve loaded it here explicitly as a reminder:

\usepackage{tracklang}

Once the resource file has been loaded, I need to iterate over all the defined entries and check if the translation field has been set. If it has, then the first language tag in the language field will supply the language, but this needs to be converted from the \texttt{ietf} language tag to a language name recognised by \polyglossia{}.

Iterating over all entries can be done with \texttt{\forglsentries} but remember that no entries will be defined before \texttt{bib2gls} has been run, so this does nothing on the first \TeX{} run.
\forglsentries{\thislabel}{% 
\glsxtrifhasfield{translation}{\thislabel}% 
{% 
  \% requires glossaries-extra v1.24 
  \glsxtrforcsvfield{\thislabel}{language}{\addfirstlang}% 
} 
%
%
}
Within the outer (\forglsentries) loop, there’s a check for the translation field using \glsxtrifhasfield. If it’s present, then the first element of the language field is required. The simplest way to get this is to use \glsxtrforcsvfield which iterates over all elements of the given field (language in this case) and break out of the loop (with \glsxxtrendfor) once the language has been found.

The handler function (\addfirstlang) is defined so that it adds the given language tag as a tracked language using \TrackLocale. This command sets \TrackLangLastTrackedDialect to the associated (tracklang) dialect label for convenience. This dialect label can then be converted to the root language label using \TrackedLanguageFromDialect. If this language is supported by polyglossia, then there should be a file called gloss-⟨language⟩.ldf.

Some of the entries use the same language, so it’s necessary to check if the language has already been defined before loading it. There’s also a problem in that the language file should not be loaded in a scoped context, but both \glsxtrforcsvfield and the unstarred \glsxtrifhasfield add implicit grouping. To solve both problems, an internal etoolbox list is defined:

\newcommand{\langlist}{}
and \xifinlist is used to first check if the language label is already in the list before adding it. Since this part of the code is scoped, the global \listxadd is used to add the language label to the list.

Next the useri field is set to text⟨language⟩ which is the name of the control sequence used with polyglossia to switch language for a short block of text. This means that \glsxtr-entryfmt{⟨text⟩} can be used to format ⟨text⟩ in the relevant language. Finally, \glsxtr-endfor is used to break out of the loop.

\newcommand*{\addfirstlang}{[1]{% 
  \TrackLocale[#1]\
  \edef\thislanguage{% 
    \TrackedLanguageFromDialect\TrackLangLastTrackedDialect% 
  } 
  \IfFileExists{gloss-\thislanguage.ldf}{% 
    {% 
      \xifinlist{\thislanguage}{\langlist}{\} 
      {\listxadd{\langlist}{\thislanguage}} 
      \xGlsXtrSetField{\thislabel}{useri}{text\thislanguage} 
    } 

519
Once the `{forglstentries}` loop has found the appropriate languages, it’s now necessary to iterate over the internal list `{langlist}` and set the language:

\forlistloop{\setotherlanguage}{\langlist}

The long-short-user style now needs to be adjusted to ensure that it picks up the appropriate language change. By default this style checks the `useri` field, so this needs to be changed to `translation` by redefining `{glsxtruserfield}`:

\renewcommand*{\glsxtruserfield}{translation}

The command that governs the format of the parenthetical material (`{glsxtruserparen}`) also needs adjusting. I’ve changed the space before the parenthesis to `{\␣}` because some of the long fields end with a full stop and this corrects the spacing. The `translation` field is in English, so this needs to be encapsulated with `{\textenglish}` in case the surrounding text is in a different language.

\renewcommand*{\glsxtruserparen}[2]{% \␣ \glsxtrparen{#1% \ifglshasfield{\glsxtruserfield}{#2}{, \textenglish{\glscurrentfieldvalue}}{}}}%

Next I’ve defined a convenient command for use in the `textformat` attributes for the custom `tug` category:

\newcommand*{\tugtextformat}[1]{% \glsxtrentryfmt{\glslabel}{#1}% }

This uses `{\glsxtrentryfmt}` to encapsulate the given text in the appropriate language command (if provided). When this is set as the `textformat` attribute, it will be used instead of `{\glstextformat}`, which means that the entry label can be referenced with `{\glslabel}`.

There’s a similar command for use in the `glossnamefont` attribute. This is used in the glossary, so the label is referenced with `{\glscurrententrylabel}`:

\newcommand*{\tugnameformat}[1]{% \glsxtrentryfmt{\glscurrententrylabel}{#1}% }

The attributes can now be set to the relevant control sequence name:
The document uses the bookindex style, which is set in the package options:

\usepackage[record, nostyles, stylemods={bookindex}, style={bookindex}]{glossaries-extra}

The bookindex style ignores the description field, so I've provided a post-name hook to append it in parentheses (with the translation, if provided):

\newcommand{\glsxtrpostnametug}{% 
  \ifglishasdesc{\glscurrententrylabel} \% 
  {\\ }\{\glossentrydesc{\glscurrententrylabel}\% 
    {, , \textenglish{\glscurrentfieldvalue}}\% 
  \} \% 
  \% 
  {\glsxtrifhasfield{translation}{\glscurrententrylabel}\% 
    {\\ }\{\textenglish{\glscurrentfieldvalue}\% 
      \} \% 
  } \% 
}

Remember that this hook is included within the name font (provided by the glossnamefont attribute in this case) so \textenglish is again used to switch the language to English for the translation.

The complete document code is listed below. The document build is:

xelatex sample-usergroups
bib2gls --group sample-usergroups
xelatex sample-usergroups
xelatex sample-usergroups
xelatex sample-usergroups

The two pages of the document are shown in figure 8.18. Since the entries have all been referenced on page 1, the location lists are all simply “1”.

\documentclass{scrreprt}
\usepackage{fontspec}
\setmainfont{Linux Libertine O}
\usepackage{polyglossia}
8 Examples: sample-usergroups.tex

\setmainlanguage[variant=uk]{english}
\usepackage{tracklang}
\usepackage{etoolbox}

\usepackage[record,% use bib2gls
nostyles,% don't load default styles
stylemods={bookindex},
style={bookindex}
]{glossaries-extra}

\glsaddstoragekey{language}{}{\glsentrylanguage}
\glsaddstoragekey{translation}{}{\glsentrytranslation}

\setabbreviationstyle[tug]{long-short-user}
\GlsXtrLoadResources[
  src={usergroups}, % data in usergroups.bib
  check-end-punctuation={name,short},
  category=tug
]

\renewcommand*{\glsxtrifcustomdiscardperiod}[2]{%
  \ifgls hasshort{\glslabel}%
    {%
      \glsxtrifwasfirstuse{}%
      {%
        \GlsXtrIfFieldUndef{shortendpunc}{\glslabel}{#2}{#1}%
      }%
    }%
  {%
    \GlsXtrIfFieldUndef{nameendpunc}{\glslabel}{#2}{#1}%
  }%
%
}
\newcommand{\langlist}{%}
\newcommand*{\addfirstlang}[1]{%
  \TrackLocale{#1}%
  \edef\thislanguage{%
    \TrackedLanguageFromDialect\TrackLangLastTrackedDialect}%
  \IfFileExists{gloss-\thislanguage.ldf}%
  {%
    \xifinlist{\thislanguage}{\langlist}{%}
    {\listxadd{\langlist}{\thislanguage}%%
     \xGlsXtrSetField{\thislabel}{\useri}{text\thislanguage}%
    \glsxtrendfor
  }%
}
8 Examples: sample-usergroups.tex

\%  
{\%}
)
\forcglslentries{\thislabel}{\%  
  \glsxtrifhasfield{translation}{\thislabel}{\%  
    \% requires glossaries-extra v1.24  
    \glsxtrforcsvfield{\thislabel}{language}{\addfirstlang}\%  
  }\%  
}{\%}
\forlistloop{\setotherlanguage}{\langlist}
\renewcommand{\glsxtrusercontentfield}{translation}
\renewcommand{\glsxtrusercontentparen}[2]{\%  
  \glsxtrparen{#1  
    \ifglshasfield{\glsxtrusercontentfield}{#2}{,  
      \textenglish{\glscurrentfieldvalue}}}{}\%  
}\%  
\newcommand{\tugtextformat}[1]{\%  
  \glsxtrentryfmt{\glslabel}{#1}\%  
}\%  
\newcommand{\tugnameformat}[1]{\%  
  \glsxtrentryfmt{\glscurrententrylabel}{#1}\%  
}\%  
\glssetcategoryattribute{tug}{textformat}{tugtextformat}\%  
\glssetcategoryattribute{tug}{glossnamefont}{tugnameformat}\%  
\newcommand{\glsxtrpostnametug}{\%  
  \ifglshasdesc{\glscurrententrylabel}{\%  
    (\glossentrydesc{\glscurrententrylabel}\%  
      \glsxtrifhasfield{translation}{\glscurrententrylabel}{\%  
        {, \textenglish{\glscurrentfieldvalue}}}{}\%  
    )\%  
  }\%  
}\%  
{\%  
  \glsxtrifhasfield{translation}{\glscurrententrylabel}{\%  
    {\textenglish{\glscurrentfieldvalue}}}\%  
}\%  
{523}
8 Examples: sample-multi1.tex

\begin{document}
\chapter{Sample}
\section{First Use}
\gls{TUG}. \gls{bgTeX}. \gls{latex-br}. \gls{CTeX}. \gls{CSTUG}. \gls{DANTE}. \gls{DKTUG}. \gls{EUG}. \gls{CervanTeX}. \gls{TirantloTeX}. \gls{GUTenberg}. \gls{UKTUG}. \gls{MaTeX}. \gls{ITALIC}. \gls{ÍsTeX}. \gls{GuIT}. \gls{KTS}. \gls{LTVG}. \gls{mxTeX}. \gls{NTG}. \gls{NTUG}. \gls{GUST}. \gls{GUTpt}. \gls{VietTUG}. \gls{LUGSA}.
\section{Next Use}
\gls{TUG}. \gls{bgTeX}. \gls{latex-br}. \gls{CTeX}. \gls{CSTUG}. \gls{DANTE}. \gls{DKTUG}. \gls{EUG}. \gls{CervanTeX}. \gls{TirantloTeX}. \gls{GUTenberg}. \gls{UKTUG}. \gls{MaTeX}. \gls{ITALIC}. \gls{ÍsTeX}. \gls{GuIT}. \gls{KTS}. \gls{LTVG}. \gls{mxTeX}. \gls{NTG}. \gls{NTUG}. \gls{GUST}. \gls{GUTpt}. \gls{VietTUG}. \gls{LUGSA}.
\printunsrtglossaries
\end{document}

sample-multi1.tex

This example uses bacteria.bib, markuplanguages.bib, vegetables.bib, minerals.bib, animals.bib, chemicalformula.bib, baseunits.bib and derivedunits.bib. Since there’s one or more UTF-8 character, the document requires UTF-8 support:

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

The aim of this example document is to have a separate glossary (without number lists) for each type of data (bacteria, markup languages, vegetables, minerals, animals, chemical formula, base units and derived units) and also an index listing all referenced entries with number lists as well as aliased entries that haven’t explicitly been used but the cross-reference term as been indexed. This requires:

selection={recorded and deps and see}

to ensure the aliased entries are selected.
Since I don’t need the default main glossary (I’m providing my own custom glossaries) I’ve used the \nomain option to suppress its automatic creation, but I do want the index glossary so I’ve used the index package option. As with the other examples, I’ve used \nostyles to suppress the creation of the default styles and used stylemods to load the particular style packages that I need and use glossaries-extra-stylemods to patch them. The index needs to be in an unnumbered chapter, which is the default for book-like styles, but I want the other glossaries in unnumbered sections so I’ve used the section option. I just need to remember to switch this before displaying the index:

\usepackage[record,% use bib2gls section,% use \section* for glossary headings postdot,% insert dot after descriptions in glossaries nomain,% don't create 'main' glossary index,% create 'index' glossary nostyles,% don't load default styles % load and patch required style packages: stylemods={list,mcols,tree,bookindex} ]{glossaries-extra}

The remaining glossaries need defining:

\newglossary{bacteria}{Bacteria}
\newglossary*{markuplanguage}{Markup Languages}
\newglossary*{vegetable}{Vegetables}
\newglossary*{mineral}{Minerals}
\newglossary*{animal}{Animals}
\newglossary*{chemical}{Chemical Formula}
\newglossary*{baseunit}{SI Units}
\newglossary*{derivedunit}{Derived Units}

As with sample-bacteria.tex and sample-markuplanguages.tex I need to set the abbreviation styles before the abbreviations are defined:

\setabbreviationstyle[bacteria]{long-only-short-only}
\setabbreviationstyle[markuplanguage]{long-short-desc}

Unlike the sample-markuplanguages.tex example, I’m not interested in tagging the initials in this case, but I still want to change the way the name field is set with the long-short-desc abbreviation style:

\renewcommand*{\glsxtrlongshortdescname}{%
\protect\glsabbrvfont{\the\glsshorttok}\space
\glsxtrparen{\glslongfont{\the\glslongtok}}%
}

Remember that this also needs to be set before the abbreviations are defined. The textformat and glossnamefont attributes may be set after definition:

\newcommand{\bacteriafont}[1]{\emph{#1}}
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont}

The description font also needs to be set since this will contain the long form:

\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont}

The markuplanguage glossary contains descriptions and some long names, so it’s better suited to the altlist style, in which case the descriptions would look better if they started with a capital letter:

\glssetcategoryattribute{markuplanguage}{glossdesc}{firstuc}

Remember that the altlist style uses the description environment, which is governed by the document class (and may be modified by list-related packages). In this case, one of the KOMA-Script classes is used, so the list items are typeset in sans-serif.

There are various ways of dealing with the duplicated data in the index, such as using the secondary option or having a separate resource set with a copy action. In this case, I’ve decided to use a dual entry system. Since the entries aren’t defined using any dual types, I’ve used entry-type-aliases to make bib2gls treat them as though they were, and I also need to alias the custom @chemical, @unit and @measurement entry types:
8 Examples: sample-multil.tex

entry-type-aliases={
  abbreviation=dualindexabbreviation,
  entry=dualindexentry,
  symbol=dualindexsymbol,
  unit=dualindexsymbol,
  measurement=dualindexsymbol,
  chemical=dualindexsymbol
}

Note that I haven’t aliased the @index types as I only want these in the index and not replicated in a separate glossary.

The primary entries for the @dualindexabbreviation type ignore the short form. It would be useful to store it. This could be done by copying the short field with replicate-fields. For example, replicate-fields={short=symbol}. However, this will cause the symbol field to be set for both the primary and dual entries, which will cause an unwanted duplication if the dual entries are displayed using a glossary style that shows the symbol field. Another field (such as user1) could be used instead or \bibglsnewdualindexabbreviation could be defined before \GlsXtrLoadResources:

\newcommand{\bibglsnewdualindexabbreviation}[7]{%
  \longnewglossaryentry*{#1}{%
    name={\protect\bibglsuselongfont{#4}{\glscategory{#2}}},%
    symbol={\protect\bibglsuseabbrvfont{#5}{\glscategory{#2}}},%
    category={index},#3}{}%
}

However, this will affect all @dualindexabbreviation entry types, but it’s not necessary for the bacteria abbreviations. Instead it’s simpler to just keep a record of the dual label so that the short form can be obtained from the dual entry:

dual-field

By default, the @dualindexabbreviation entry type falls back on the short field if the name is omitted. In this case I want it to fall back on the long field instead.

abbreviation-name-fallback={long}

Remember that the sort fallback for abbreviations is still short (but can be changed with abbreviation-sort-fallback), but I’ve changed the sort fallback for symbols:

symbol-sort-fallback={name}

I also need to alias the custom fields (especially for those in the chemicalformula.bib, baseunits.bib and derivedunits.bib files):

field-aliases={
  identifier=category,
8 Examples: sample-multil.tex

\begin{verbatim}
formula=symbol,
chemicalname=name,
unitname=name,
unitsymbol=symbol,
measurement=description
\}

There's a slight problem here. This ensures that the entries defined in chemicalformula.bib have a name and symbol field, which are swapped round for the dual (according to the default dual-indexsymbol-map) but these entries don't have a description field. Since I'd like to use the mcolalttreegroup style, this will end up with the odd appearance of the formula (stored in the name field for the dual) followed by the chemical name (stored in the symbol field for the dual) in parenthesis. This is the default ⟨name⟩ (⟨symbol⟩) ⟨description⟩ format for the style. I've fixed this by locally redefining \glsxtralttreeSymbolDescLocation for just that glossary:

\printunsrtglossary*[type={chemical},style={mcolalttreegroup}]
{% 
\renewcommand\glsxtralttreeSymbolDescLocation[2]{% 
\glossentrysymbol{#1}\glspostdescription\glsxtrAltTreePar
}% 
\renewcommand*{\glstreenamefmt}[1]{#1}% 
\renewcommand*{\glstreegroupheaderfmt}[1]{\textbf{#1}}%
}

I've also redefined \glstreenamefmt to prevent the names appearing in bold, which means I also need to redefine \glstreegroupheaderfmt to keep the headers bold.

All the @dualindex\{type\} entry types provide a primary entry that behaves like @index. The secondary behaves like @\{type\}. This means that the primaries are conveniently gathered together with all the unaliased @index entries, so the primary entry type needs to be set to index:

type={index}

The dual entry type depends on the entry's category. Since I've defined my custom glossaries with a label that matches the custom identifier field, I can both alias this custom field to the category field and also set dual-type so that it matches the category:

field-aliases={identifier=category},
dual-type={same as category}

The primary entries (in the index glossary) need to be sorted alphabetically, and since the document is in English I'm sorting according to that language (identified by the language code en), but I also want to make sure that all the primary entries are sorted by the name field to avoid discrepancies in the fallback value for the sort field:

sort={en},
sort-field={name}
\end{verbatim}
With `abbreviation-name-fallback={long}` now set, this means that *Coxiella burnetii* comes after *Clostridium tetani* in the index. I haven’t changed the sort field for the dual entries, so in that case the `abbreviation-sort-fallback` and `symbol-sort-fallback` settings will be used with the duals. This means that *C. burnetii* is between *C. botulinum* and *C. perfringens* rather than after *C. tetani*.

I’d like to sort the dual entries according to a letter-number rule (as for the above `sample-chemical.tex` and `sample-units3.tex` examples) but this would order “bilinite” after “biotite” in the minerals glossary, so instead I’m also using the English sort rule for the duals, but with the numbers padded:

```
dual-sort={en},
dual-sort-number-pad={2},
```

This method doesn’t work as well as the method used in `sample-chemical.tex` as it doesn’t separate the capitals, digits and lower case characters in the way that can be achieved with the letter-number methods. An improvement can be made by changing the break-points. I could use `dual-break-at={upper-upper}` but this would put “seal” before “sea lion” in the animal glossary, so instead I’ve used:

```
dual-break-at={upper-upper-word}
```

This now puts “sea lion” before “seal”. Unfortunately the word break points will cause a break at the markers used to indicate positive and negative numbers that are inserted with `dual-sort-number-pad`, so these need to be changed to something that won’t cause them to be discarded:

```
dual-sort-pad-minus={0},
dual-sort-pad-plus={1}
```

The document loads `hyperref` which means that all the \gls{svg} references will create hyperlinks. Since the primaries are in the index, the default prefixes mean that, for example, \gls{svg} links to the “scalable vector graphics” item in the index rather than to the abbreviation “SVG” in the markup language glossary. There are two alternatives: change \gls{svg} to \gls{dual.svg} or change the default prefixes, which is the more convenient approach and is the one used here:

```
label-prefix={idx.},

dual-prefix={}
```

Now \gls{svg} refers to the dual abbreviation “SVG” and \gls{idx.svg} refers to the primary entry “scalable vector graphics”. Unfortunately this means that the records created with \gls{svg} now refer to the dual abbreviation and will end up being displayed in the glossary instead of the index. This can be fixed with:

```
combine-dual-locations={primary}
```
Which transfers the dual entry locations to the corresponding primary.

The other problem is the cross-references in the description fields. Since the labels don’t start with “dual.” bib2gls will assume they refer to the primary entries, which means that “idx.” (the value of label-prefix) will be inserted. This means that they’ll link to the index rather than the glossary entry. It also means that the cross-references where the dual is an abbreviation won’t behave like an abbreviation as the reference is to the primary (non-abbreviation) entry. This can be fixed by setting cs-label-prefix to the same value as dual-prefix:

cs-label-prefix={}

The index is displayed using the bookindex style. This doesn’t show the description or symbol by default, but it would be useful to include the symbol in parentheses after the name. This can be done by redefining \glsxtrbookindexname:

\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%,
  \ifglsifcategory{#1}{chemical}{, \glossentrysymbol{#1}}%
  \{}%
}\%

However the chemical forumlae look a little odd in parentheses (especially those that contain parenthetical parts) but this can be fixed by adding a category check:

\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%,
  \ifglsifcategory{#1}{chemical}{%
    {, \glossentrysymbol{#1}}%
    \{}%
  }%
}\%

Unfortunately \glossentrysymbol doesn’t pick up the glossnamefont attribute, so if the short form of the abbreviations is saved in the symbol field, using one of the methods discussed above, then the custom \bacteriafont won’t be applied. (As from glossaries-extra version 1.42, there is now the glosssymbolfont attribute that’s used by \glossentrysymbol.)

A simple solution is to use \glossentrynameother instead:

\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%,
  \ifglsifcategory{#1}{chemical}{%
    {, \glossentrysymbol{#1}}%
  }%
}\%
However, since I decided not to store the short form in the \texttt{symbol} field and just saved the dual entry label instead, I need to lookup the short form from the dual entry:

\renewcommand*{\glxtrbookindexname}{[1]}
\glossentryname{#1}
\ifglschassymbol{#1}
  \glsifcategory{#1}{chemical}
  , \glossentrysymbol{#1}
  \glossentrynameother{#1}{symbol}
\else
  \glsifcategory{#1}{markuplanguage}
  \glsxtrifhasfield{short}{\glsxtrusefield{#1}{dual}}
  \glossentrynameother{#1}{symbol}
\fi

Not all of the markup languages are abbreviations so this uses \texttt{\textbackslash glsxtrifhasfield} to check if the \texttt{short} field is set. The dual entry’s label is easily obtained because \texttt{\textbackslash dual-field} has provided the \texttt{dual} internal field and set it to the corresponding label.

It’s sometimes useful for the index to include a reference to the term’s definition. This can be done by making use of \texttt{\textbackslash glsextrapostnamehook}, which can be redefined before the glossaries to automatically record each entry:

\renewcommand{\glsextrapostnamehook}{[1]\glsadd[format={hyperbf}]{#1}}

This needs to be redefined to ignore its argument before the index, to avoid the redundant index record:

\renewcommand{\glsextrapostnamehook}{[1]}

Remember that if any records are added within a glossary, an extra \texttt{\textbackslash \textbackslash pdflatex} and \texttt{\textbackslash \textbackslash bib2gls} call are required to ensure that the location list is correct, so the document build is:

pdflatex sample-multi1
bib2gls --group sample-multi1
pdflatex sample-multi1
bib2gls --group sample-multi1
pdflatex sample-multi1

531
The complete document code is listed below. The resulting document is shown in figure 8.19 and figure 8.20.

\documentclass{scrreprt}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[version=4]{mhchem}
\usepackage{siunitx}
\usepackage[colorlinks]{hyperref}
\usepackage[record,% use bib2gls
section,% use \section* for glossary headings
postdot,% insert dot after descriptions in glossaries
nomain,% don't create 'main' glossary
index,% create 'index' glossary
nostyles,% don't load default styles
% load and patch required style packages:
\setglossariesextra{list,mcols,tree,bookindex}
]{glossaries-extra}
\newglossary*[bacteria]{Bacteria}
\newglossary*[markuplanguage]{Markup Languages}
\newglossary*[vegetable]{Vegetables}
\newglossary*[mineral]{Minerals}
\newglossary*[animal]{Animals}
\newglossary*[chemical]{Chemical Formula}
\newglossary*[baseunit]{SI Units}
\newglossary*[derivedunit]{Derived Units}

% abbreviation styles must be set before \GlsXtrLoadResources:
\setabbreviationstyle[bacteria]{long-only-short-only}
\setabbreviationstyle[markuplanguage]{long-short-desc}

% style-dependent name format must be set
% before \GlsXtrLoadResources:
\renewcommand*\glsxtrlongshortdescname{%
  \protect\glsabbrvfont{\the\glsshorttok}\space
  \glsxtrparen{\glslongfont{\the\glslongtok}}%
}
\GlsXtrLoadResources[
src={bacteria,markuplanguages,vegetables,minerals,animals,chemicalformula,baseunits,derivedunits},
selection={recorded and deps and see},
set-widest,
type=index,
\newcommand{\bacteriafont}[1]{\textit{#1}}
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont}
\glssetcategoryattribute{markuplanguage}{glossdesc}{firstuc}
\renewcommand*{\glsxtrbookindexname}[1]{\% \glossentryname{#1}\% \ifglsishassymbol{#1}\% \% \glsifcategory{#1}{chemical}\% \{, \glossentrysymbol{#1}\% \} \}}

533
\begin{document}
\chapter{Sample}
\section{Bacteria}
\subsection{First Use}
\gls{cbotulinum}, \gls{pputida}, \gls{cperfringens}, \gls{bsubtilis}, \gls{ctetani}, \gls{pcomposti}, \gls{pfimeticola}, \gls{cburnetii}, \gls{raustralis}, \gls{rrickettsii}.
\subsection{Next Use}
\gls{cbotulinum}, \gls{pputida}, \gls{cperfringens}, \gls{bsubtilis}, \gls{ctetani}, \gls{pcomposti}, \gls{pfimeticola}, \gls{cburnetii}, \gls{raustralis}, \gls{rrickettsii}.

\section{Markup Languages}
\subsection{First Use}
\gls{LaTeX}, \gls{markdown}, \gls{xhtml}, \gls{mathml}, \gls{svg}.
\subsection{Next Use}
\gls{LaTeX}, \gls{markdown}, \gls{xhtml}, \gls{mathml}, \gls{svg}.

\section{Vegetables}
\gls{cabbage}, \gls{brussels-sprout}, \gls{artichoke}, \gls{cauliflower}, \gls{courgette}, \gls{spinach}.

\section{Minerals}
\gls{beryl}, \gls{amethyst}, \gls{chalcedony}, \gls{aquamarine}, \gls{aragonite}, \gls{calcite}, \gls{bilinite}, \gls{cyanotrichite}, \gls{biotite}, \gls{dolomite}, \gls{quetzalcoatlite}, \gls{vulcanite}.

\section{Animals}

534
\section{Chemicals}
\gls{duck}, \gls{parrot}, \gls{hedgehog}, \gls{sealion}.

\section{SI Units}
Base: \gls{ampere}, \gls{kilogram}, \gls{metre}, \gls{second}, \gls{kelvin}, \gls{mole}, \gls{candela}.
Derived: \gls{area}, \gls{volume}, \gls{velocity}, \gls{acceleration}, \gls{density}, \gls{luminance}, \gls{specificvolume}, \gls{concentration}, \gls{wavenumber}.

\chapter*{Glossaries}
\renewcommand{\glsextrapostnamehook}[1]{}\printunsrtglossary[type=bacteria,style=mcoltree]
\printunsrtglossary[type=markuplanguage,style=altlist]
\printunsrtglossary[type=vegetable,style=tree,nogroupskip]
\printunsrtglossary[type=mineral,style=treegroup]
\printunsrtglossary[type=animal,style=tree]
\printunsrtglossary*[type=chemical,style=mcolalttreegroup]{%\renewcommand\glsxtralttreeSymbolDescLocation[2]{%\glossentrysymbol{#1}\glspostdescription\glsxtrAltTreePar\}%),\renewcommand*{\glstreenamefmt}[1]{\textbf{#1}}\renewcommand*{\glstreegroupheaderfmt}[1]{textbf{#1}}%}\printunsrtglossary[type=baseunit,style=alttree]
\printunsrtglossary[type=derivedunit,style=alttree]\renewcommand{\glsextrapostnamehook}[1]{}\setupglossaries{section=chapter}\printunsrtglossary[type=index,style=bookindex}\end{document}

\textbf{sample-multi2.tex}

This example is an alternative approach to sample-multi1.tex. Instead of using dual entry types to define entries that appear in both a glossary and the index, this example makes use of \texttt{record-label-prefix} to reselect the recorded entries for the index. This is more
### 1 Sample

#### 1.1 Bacteria

1.1.1 First Use

Clostridium botulinum, Pseudomonas putida, Thiomicrospira denitrificans, Bacillus subtilis, Clostridium tetani, Porphyromonas gingivalis, Francisella tularensis, Rickettsia rickettsii, Planispora verrucosa, Rickettsia prowazekii, Actinobacillus rinositis.

1.1.2 Next Use


#### 1.2 Markup Languages

1.2.1 First Use

TX, markdown, extensible hypertext markup language (XHTML), mathematical markup language (MathML), scalable vector graphics (SVG).

1.2.2 Next Use

TX, markdown, MathML, SVG.

### 1.3 Vegetables

- Broccoli, Brussels sprout, artichoke, cauliflower, courgette, spinach.

### 1.4 Minerals

- Beryl, amethyst, chalcedony, aquamarine, aragonite, calcite, bílinite, cyanotrichite, biotite, dolomite, quartz, quetzalcoatlite, vulcanite.

### 1.5 Animals

Duck, parrot, hedgehog, sea lion, otite, dolomite, quetzalcoatlite, vulcanite.

### 1.6 Chemicals

- Al₂(SO₄)₃, H₂O, C₆H₁₂O₆, CH₃CH₂OH, CH₂O, OF₂, O₂F₂, SO₄²⁻, H₃O⁺, OH⁻, O₂, H₂O, AlF₃, O, Al₂CoO₄, As₄S₄, ... SI Units

#### Base

A, kg, m, s, K, mol, cd. Derived: m², m³, m s⁻¹, m s⁻², A m⁻², cd m⁻², m³ kg⁻¹, mol m⁻³, m⁻¹.

#### 1.7 SI Units

Base: A, kg, m, s, K, mol, cd. Derived: m², m³, m s⁻¹, m s⁻², A m⁻², cd m⁻², m³ kg⁻¹, mol m⁻³, m⁻¹.
### Derived Units

- **kg** (kilogram) mass.
- **m** (metre) length.
- **s** (second) time.
- **Ampere** (ampere) electric current.
- **cd** (candela) luminous intensity.
- **mol m⁻³** (mole per cubic metre) concentration.
- **kJ kg⁻¹** (kilojoule per kilogram) specific volume.
- **mol (mole) amount of substance.
- **s⁻¹** (second) frequency.

### SI Units

- **m** (metre) length.
- **kg** (kilogram) mass.
- **s** (second) time.
- **Ampere** (ampere) electric current.
- **cd** (candela) luminous intensity.
- **mol m⁻³** (mole per cubic metre) concentration.
- **kJ kg⁻¹** (kilojoule per kilogram) specific volume.
- **mol (mole) amount of substance.

### Chemical Formula

- **H₂O** water.
- **NaCl** sodium chloride.
- **CaCO₃** calcium carbonate.
- **CO₂** carbon dioxide.
- **CH₄** methane.
- **H₂** hydrogen.
- **O₂** oxygen.

### Index

- **A** ammonium nitrate, NH₄NO₃
- **B** benzene, C₆H₆
- **C** carbon, C
- **D** diesel fuel, C₁₀H₈N
- **E** ethanol, C₂H₅OH
- **F** formaldehyde, CH₂O
- **G** gasoline
- **H** hydrogen, H₂
- **K** potassium, K
- **M** magnesium, Mg
- **N** nitrogen, N
- **O** oxygen, O
- **P** petrol, C₈H₁₈
- **Q** quartz
- **R** rubber, C₃H₅(OH)₃
- **S** sulfur, S
- **T** tungsten, W
- **V** vanadium, V
- **Z** zinc, Zn

---

**Figure 8.20: sample-multi1.pdf (pages 5 to 8)**
complicated but it allows the entries that have natural word ordering to use a locale sort method while the entries that are symbolic can use one of the letter-number sort methods.

This document uses some additional .bib files to the previous example, so it has extra glossaries, which all need to be defined:

\newglossary*{bacteria}{Bacteria}
\newglossary*{markuplanguage}{Markup Languages}
\newglossary*{vegetable}{Vegetables}
\newglossary*{mineral}{Minerals}
\newglossary*{animal}{Animals}
\newglossary*{chemical}{Chemical Formula}
\newglossary*{baseunit}{SI Units}
\newglossary*{measurement}{Measurements}
\newglossary*{film}{Films}
\newglossary*{book}{Books}
\newglossary*{person}{People}
\newglossary*{mediacontrol}{Media Control Symbols}
\newglossary*{information}{Information Symbols}
\newglossary*{weather}{Weather Symbols}

Note that this is a total of 15 glossaries (including the index). With the basic \makeglossaries method, this would require 16 write registers (including the write register used to create the indexing style file), and a total of $15 \times 3 + 1 = 46$ associated files. (This doesn’t include the standard .aux file and the .out file created by hyperref.) With bib2gls, no additional write registers are required and the number of associated bib2gls files is equal to the number of resource commands plus the transcript file (in this example, $9 + 1 = 10$).

Since this document requires people.bib, books.bib and films.bib it also requires the files that supply the definitions of the custom commands (no-interpret-preamble.bib and either interpret-preamble.bib or interpret-preamble2.bib) to ensure the custom commands are provided both for the document and for bib2gls’s interpreter.

The first resource set to be loaded simply reads no-interpret-preamble.bib with the preamble interpreter switched off:

\GlsXtrLoadResources[
  src={no-interpret-preamble},
  interpret-preamble={false}
]

This ensures that Bib\TeX{} can pick up the provided commands and prevents them from being added to the interpreter.

The people.bib file is the next to be loaded with interpret-preamble.bib. This is loaded separately from the other resources as this needs the name field to be copied to first (if not already set), as in the sample-people.tex file. By having a separate resource set, this setting doesn’t affect the other entries. I’ve also converted the date fields so that I can customise the format in the document.
\GlsXtrLoadResources[
  src={interpret-preamble,people},
  field-aliases={
    identifier=category,
    born=user1,
    died=user2,
    othername=user3
  },
  replicate-fields={name={first}},
  type={person},
  save-locations={false}
  date-fields={user1,user2},
  date-field-format={d MMM y G}
]

As with the sample-people.tex document, I need to use the --break-space switch to convert the ~ to a normal breakable space so that it matches the given format. I’ve loaded the datetime2 package:\footnote{The en-GB option to datetime2 also requires that datetime2-english must be installed.}

\usepackage[en-GB]{datetime2}

so that I can use \DTMdisplaydate to adjust the formatting:

\newcommand*{\bibglsdate}[7]{\DTMdisplaydate{#1}{#2}{#3}{#4}}

This needs to go before the resource set is loaded. Note that the en-GB option identifies the document locale as en-GB (since there are no language packages loaded).

Note that unlike sample-people.tex which had category={people}, this document obtains the category field from the custom identifier field, which in this case has the value person. This means that the category hooks from sample-people.tex need to be renamed to reflect the different category label:

\newcommand*{\glsxtrpostlinkperson}{%\glsxtrifwasfirstuse
  {%
    \ifglsfinalfield{user3}{\glslabel}
    {\space(\glscurrentfieldvalue)}
  {}}
%
}

\newcommand*{\glsxtrpostnameperson}{%\glsxtrifwasfirstuse
  {%
    \ifglsfinalfield{user3}{\glscurrententrylabel}
    {\space(\glscurrentfieldvalue)}
  {}}
%}
\newcommand*{\glsxtrpostdescperson}{% 
\ifglshasfield{user1}\{\glscurrententrylabel\} 
{\% born 
\space(\glscurrentfieldvalue,--\,%, 
\\ifglshasfield{user2}\{\glscurrententrylabel\} 
{\% died 
\glscurrentfieldvalue 
\} 
{\% 
\} 
}}} 
{\% 
}}} 

The other .bib files that require locale sorting can now be loaded, but remember that the abbreviation style settings must be set first since this resource set includes abbreviations:

\setabbreviationstyle[bacteria]{long-only-short-only} 
\setabbreviationstyle[markuplanguage]{long-short-desc} 

\renewcommand*{\glsxtrlongshortdescname}{% 
\protect\glsabbrvfont{the}\glsshorttok\space 
(TeX Users Group)} 

Now the resource set can be loaded:

\GlsXtrLoadResources[ 
src={bacteria,markuplanguages,vegetables, 
minerals,animals,books,films}, 
field-aliases={identifier=category}, 
type={same as category}, 
save-locations={false} 
] 

The semantic markup command and attributes are as for sample-multi1.tex:

\newcommand{\bacteriafont}[1]{\emph{#1}} 
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont} 
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont} 
\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont} 
\glssetcategoryattribute{markuplanguage}{glossdesc}{firstuc} 

Similarly for the books:
\newcommand{\bookfont}[1]{{\emph{#1}}}  \\
glssetcategoryattribute{book}{textformat}{\bookfont}  \\
glssetcategoryattribute{book}{glossnamefont}{\bookfont}

(as for sample-media.tex) and for films:
\newcommand{\filmfont}[1]{{\emph{#1}}}  \\
glssetcategoryattribute{film}{textformat}{\filmfont}  \\
glssetcategoryattribute{film}{glossnamefont}{\filmfont}

Next come the chemical formulae:
\GlsXtrLoadResources[  
src={chemicalformula},  
entry-type-aliases={chemical=symbol},  
field-aliases={  
  identifier=category,  
  formula=name,  
  chemicalname=description  
},  
type={chemical},  
set-widest,  
sort={letternumber-case},  
symbol-sort-fallback={name},  
save-locations={false}  
]

and the SI units, which are now combined into a single glossary:
\GlsXtrLoadResources[  
src={baseunits,derivedunits},  
entry-type-aliases={measurement=symbol,unit=symbol},  
field-aliases={  
  unitname=description,  
  unitsymbol=symbol,  
  measurement=name  
},  
category={measurement},  
type={measurement},  
set-widest,  
symbol-sort-fallback={name},  
save-locations={false}  
]

Here the \texttt{name} field is obtained from the custom \texttt{measurement} field. Since this contains a word, the default locale sort is appropriate. I’ve locally redefined \texttt{\glsxtralttreeSymbol-\texttt{DescLocation}} to place the symbol in parentheses after the description:
The base units are replicated in the `baseunit` glossary, this time with the `name` field obtained from the custom `unitsymbol` field. This means that I need to find a way to prevent duplicate labels. The simplest method is to use `duplicate-label-suffix`:

```latex
\GlsXtrLoadResources[  
src={baseunits},  
entry-type-aliases={unit=symbol},  
field-aliases={  
  unitname=description,  
  unitsymbol=name  
},  
category={measurement},  
type={baseunit},  
duplicate-label-suffix={.copy},  
symbol-sort-fallback={name},  
save-locations={false}
]
```

I can’t use `set-widest` here as it won’t pick up the modified label and will instead use the label from the original entry. Instead I’ve used `\glsFindWidestTopLevelName` to find it:

```latex
\printunsrtglossary*[type={baseunit},style={alttree},nogroupskip]
{\glsFindWidestTopLevelName[baseunit]}
```

The text symbols from `miscsymbols.bib` are all loaded in a single resource set, where the `type` field can be obtained from the `category`, which in turns is obtained from the custom `identifier` field. Since `bib2gls` doesn’t recognise any of the symbol commands, I’m sorting according to the `description` field. (Even if `bib2gls` could determine a Unicode value for each of the symbols, sorting by the description makes more sense in this case.)

```latex
\GlsXtrLoadResources[  
src={miscsymbols},  
field-aliases={  
  identifier=category,  
},
]"
Finally, all recorded and cross-referenced terms are needed for the index. This includes entries that have already been defined in the earlier resource sets (so a guard against duplicates is necessary) but it also includes entries from the `terms.bib` file that haven’t yet been dealt with. I’d like the index to start with a symbol group containing the icons from `miscsymbols.bib`. This needs to be dealt with separately from the rest of the index to keep them together in a single group:

\GlsXtrLoadResources[
  src={miscsymbols},
  selection={recorded no deps},
  duplicate-label-suffix={.copy},
  entry-type-aliases={icon=index},
  field-aliases={
    identifier=category,
    icondescription=symbol,
    icon=name
  },
  type={index},
  sort-field={symbol},
  group={glssymbols}
]

Since I know that there are no parents or cross-references in this set of entries I’ve used `selection={recorded no deps}` to skip the dependency checks. In this resource set, the `name` field has the symbol command (obtained from the custom `icon` field), and the `symbol` field has the symbol description (obtained from the custom `icondescription` field), which is used as the sort field. I’ve set the `group` label to `glssymbols`, which keeps all these entries in a single group and the title will be obtained from `\glssymbolsgroupname`.

Before loading the final resource set `\glsxtrlongshortdescname` needs to be changed so that the abbreviations using the long–short–desc style (that is, the abbreviations with the `category` set to `markuplanguage`) have the `name` field set to ⟨long⟩ ⟨(short)⟩:

\renewcommand*{\glsxtrlongshortdescname}{% 
  \protect\glslongfont{\the\glslongtok}\space \glsxtrparen{\glssabbrvfont{\the\glsshorttok}}% 
}
The long-only-short-only style has a similar command, but it was only introduced to glossaries-extra version 1.25:

\renewcommand*{\glsxtronlyname}\
\protect\glsabbrvonlyfont{\the\glslongtok}

The abbreviations all need to be sorted according to the long form:

abbreviation-sort-fallback={long}

The custom entry types and fields again need to be aliased

entry-type-aliases={
   chemical=index,
   measurement=entry,
   unit=dualentry,
   icon=index
},
field-aliases={
   identifier=category,
   formula=symbol,
   chemicalname=name,
   unitname=description,
   unitsymbol=symbol,
   measurement=name,
   icon=symbol,
   icondescription=name
}

The chemical formulae and icons are now defined using @index with the name field set to a word form (chemical name and icon description). This means they’re appropriate for alphabetical sorting. (Both @entry and @symbol require the description field, which is why I’ve aliased @chemical and @icon to @index here.) The custom @measurement entry type has a description field (obtained from unitname), so that’s aliased to @entry as again the name field is suitable for alphabetical sorting.

I’ve aliased @unit to @dualentry rather than @symbol as I want both the unit name and the measurement in the index and I’ve combined their location lists:

combine-dual-locations={both}

Both primary and dual entries need to go in the index glossary:

type={index},
dual-type={index}

All .bib files used in the previous resource sets are needed as well as the terms.bib file:
but this time I also want to select entries that haven’t been recorded but have a cross-reference to a recorded entry:

selection[recorded and deps and see]

Again it’s necessary to provide a way to avoid duplicate entry labels, which can be done with

duplicate-label-suffix={.copy},

as above. However, this will cause the cross-references (from the alias fields) to link to the glossary rather than the index. This may or may not confuse the reader. For consistency, it may be more suitable to have the cross-reference in the index link to the aliased entry in the index rather than in the glossary. I’ve therefore instead used:

label-prefix={idx.},
record-label-prefix={idx.},

This means that the entries defined in terms.bib need to be referenced with this prefix.

All instances of \gls will link to the original entry, so all entries except for those in the terms.bib file will link to the relevant glossary. Those in the terms.bib file will link to the index. It’s possible to disable the hyperlinks for those entries, but the reader may find it useful to jump to the index to look up other locations for that entry in the document.

To deal with the identical book and film titles, I’m again using the category to resolve identical sort values:

identical-sort-action={category}

For the people who have a first field, I’ve decided that this would be more appropriate for the index as it’s more compact than the name, so here I’ve done the reverse to earlier and copied the first field (if supplied) into the name field, but since the name field is already provided the override setting needs to be on:

replicate-override,
replicate-fields={first=name}

As with sample-people.tex I’ve provided some custom commands to make it easier to locally redefine \sortname and \sortvonname:

\newcommand*{\swaptwo}[2]{#2, #1}
\newcommand*{\swapthree}[3]{#2 #3, #1}

I’ve redefined \glsxtrbookindexname in a similar manner to sample-multi1.tex but it has some modifications:
\renewcommand*{\glsxtrbookindexname}[1]{% 
  \ifglsishassymbol{#1}% 
  {\%  
    \glsxifcategory{#1}{chemical}%  
    {\, \glossentrysymbol{#1}% 
      \space{\glossentrynameother{#1}{symbol}}%  
    }%  
  }%  
  \glsxifcategory{#1}{film}%  
  {\underline{\text{film}}}% 
  {}% 
  }% 
}\glsxtrbookindexname

This appends “(film)” to film names. I’ve chosen this method rather than using the post-name hook as I only want this in the index and not in the list of films.

For some of the entries that are referenced in the document, I’ve appended information in parentheses:

\gls{Al2SO43} (\glsdesc{Al2SO43})

This is all right for odd instances, but if this always needs to be done on first use, then it’s better to use the post-link hook, which is what I’ve done for the icons for comparison:

\newcommand*{\glsxtrpostlinkmediacontrol}{% \glsxtrpostlinkAddDescOnFirstUse \}
\newcommand*{\glsxtrpostlinkinformation}{% \glsxtrpostlinkAddDescOnFirstUse \}
\newcommand*{\glsxtrpostlinkweather}{% \glsxtrpostlinkAddDescOnFirstUse \}

I’ve also provided some custom commands to make it easier to reference entries without worrying about the prefixes:

\newcommand{\unit}{\glssymbol}
\newcommand{\measurement}{\gls}
\glsxtrnewgls{film.}{\film}

As with sample-multi1.tex, it would be useful to include the page where the entries are defined in their corresponding lists. Again this can be done by redefining the general purpose non-category post-name hook \glsextrapostnamehook:
This needs resetting before the index, since it’s redundant to record an entry in the index. This will require an extra \texttt{bib2gls}+\LaTeX{} system call as this code can’t be performed until the glossaries have been created.

The complete document code is listed below. The document build is:

\begin{verbatim}
pdflatex sample-multi2
bib2gls --group --break-space sample-multi2
pdflatex sample-multi2
bib2gls --group --break-space sample-multi2
pdflatex sample-multi2
\end{verbatim}

The resulting document is shown in figure 8.21, figure 8.22 and figure 8.23.

\documentclass{scrreprt}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[version=4]{mhchem}
\usepackage{siunitx}
\usepackage{etoolbox}
\usepackage{marvosym}
% package conflict, need to undefine conflicting commands
\undef\Sun
\undef\Lightning
\usepackage[weather]{ifsym}
\usepackage[en-GB]{datetime2}
\usepackage[colorlinks]{hyperref}
\usepackage[record,% use bib2gls
section,% use \texttt{section*} for glossary headings
postdot,% insert dot after descriptions in glossaries
nomain,% don’t create ‘main’ glossary
index,% create ‘index’ glossary
nostyles,% don’t load default styles
% load and patch required style packages:
stylemods={list,mcols,tree,bookindex}
]{glossaries-extra}
\newglossary*[bacteria]{Bacteria}
\newglossary*[markuplanguage]{Markup Languages}
\newglossary*[vegetable]{Vegetables}
\newglossary*[mineral]{Minerals}
\newglossary*{animal}{Animals}
\newglossary*{chemical}{Chemical Formula}
\newglossary*{baseunit}{SI Units}
\newglossary*{measurement}{Measurements}
\newglossary*{film}{Films}
\newglossary*{book}{Books}
\newglossary*{person}{People}
\newglossary*{mediacontrol}{Media Control Symbols}
\newglossary*{information}{Information Symbols}
\newglossary*{weather}{Weather Symbols}

\newcommand*{\bibglsdate}[7]{\DTMdisplaydate{#1}{#2}{#3}{#4}}

\GlsXtrLoadResources[
  src={no-interpret-preamble},
  interpret-preamble=false
]

\GlsXtrLoadResources[
  src={interpret-preamble,people},
  field-aliases={
    identifier=category,
    born=user1,
    died=user2,
    othername=user3
  },
  replicate-fields={name={first}},
  type=person,
  save-locations=false,
  date-fields={user1,user2},
  date-field-format={d MMM y G}
]

% Abbreviation styles must be set before the resource set
% that defines the abbreviations:
\setabbreviationstyle[bacteria]{long-only-short-only}
\setabbreviationstyle[markuplanguage]{long-short-desc}

% And also the style-dependent name format:
\renewcommand*{\glsxtrlongshortdescname}{%
  \protect\glsabbrvfont{\the\glsshorttok}\space
  \glsxtrparen{\glslongfont{\the\glslongtok}}%
}

\GlsXtrLoadResources[
  src={bacteria,markuplanguages,vegetables,
minerals, animals, books, films},
field-aliases={
    identifier=category,
    year=user1,
    cast=user2
},
type={same as category},
bibtex-contributor-fields={user2},
contributor-order={forenames},
save-locations=false
]
\GlsXtrLoadResources[
src={chemicalformula},
entry-type-aliases={chemical=symbol},
field-aliases={
    identifier=category,
    formula=name,
    chemicalname=description,
},
type={chemical},
set-widest,
sort={letternumber-case},
symbol-sort-fallback={name},
save-locations=false
]
\GlsXtrLoadResources[
src={baseunits, derivedunits},
entry-type-aliases={measurement=symbol, unit=symbol},
field-aliases={
    unitname=description,
    unitsymbol=symbol,
    measurement=name
},
category={measurement},
type={measurement},
set-widest,
symbol-sort-fallback={name},
save-locations=false
]
\GlsXtrLoadResources[
src={baseunits},
entry-type-aliases={unit=symbol},
field-aliases={

549
unitname=description, 
unisymbol=name 
},
category={measurement},
type={baseunit},
duplicate-label-suffix={.copy},
symbol-sort-fallback={name},
save-locations=false 
]

\GlsXtrLoadResources[
 src={miscsymbols},
 field-aliases={
   identifier=category, 
   icon=name, 
   icondescription=description 
},
 entry-type-aliases={icon=symbol},
type={same as category},
sort-field={description},
save-locations=false, 
set-widest 
]
\renewcommand*{\glsxtrlongshortdescname}{\%
   \protect\protect\glslongfont{\the\glslongtok}\space
   \glsxtrparen{\glsabbrvfont{\the\glsshorttok}}% 
}
% requires glossaries-extra v1.25:
\renewcommand*{\glsxtronlyname}{\%
   \protect\glsabbrvonlyfont{\the\glslongtok}% 
}

\GlsXtrLoadResources[
 src={miscsymbols},
 selection={recorded no deps},
 duplicate-label-suffix={.copy},
 entry-type-aliases={icon=index},
 field-aliases={
   identifier=category, 
   icondescription=symbol, 
   icon=name 
},
type=index, 
 sort-field={symbol}, 
}
group={glssymbols}

\GlsXtrLoadResources[
src={terms,bacteria,markuplanguages,vegetables,minerals,
animals,chemicalformula,baseunits,derivedunits,people,
films,books,miscsymbols},
selection={recorded and deps and see},
field-aliases={
  identifier=category,
  formula=symbol,
  chemicalname=name,
  unitname=description,
  unitsymbol=symbol,
  measurement=name,
  icon=symbol,
  icondescription=name
},
entry-type-aliases={
  chemical=index,
  measurement=entry,
  unit=dualentry,
  icon=index
},
label-prefix={idx.},
record-label-prefix={idx.},
type=index,
dual-type=index,
combine-dual-locations=both,
abbreviation-sort-fallback={long},
replicate-override,
replicate-fields={first=name},
identical-sort-action={category}
]

\newcommand*{\swaptwo}[2]{#2, #1}
\newcommand*{\swapthree}[3]{#2 #3, #1}

\newcommand{\bacteriafont}[1]{\emph{#1}}
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont}

\newcommand{\bookfont}[1]{\emph{#1}}
\glssetcategoryattribute{book}{textformat}{bookfont}
\glssetcategoryattribute{book}{glossnamefont}{bookfont}
\newcommand{\filmfont}{\textbf{#1}}
\glssetcategoryattribute{film}{textformat}{\textbf{}}
\glssetcategoryattribute{film}{glossnamefont}{\textbf{}}
\glssetcategoryattribute{film}{glossdesc}{\textsc{}}
\glssetcategoryattribute{markuplanguage}{glossdesc}{\textsc{}}
\newcommand{\glsxtrpostlinkmediacontrol}{\%\glsxtrpostlinkAddDescOnFirstUse\}%
\newcommand{\glsxtrpostlinkinformation}{\%\glsxtrpostlinkAddDescOnFirstUse\}%
\newcommand{\glsxtrpostlinkweather}{\%\glsxtrpostlinkAddDescOnFirstUse\}%
\newcommand{\glsxtrpostlinkperson}{\%\glsxtrifwasfirstuse\%
  \ifgls\has\field{user3}{\glslabel}\%
  \text{(\glscurrentfieldvalue)}\%
  \%
  \%
  \%
\}
\newcommand{\glsxtrpostnameperson}{\%\ifgls\has\field{user3}{\glscurrententrylabel}\%
  \text{(\glscurrentfieldvalue)}\%
  \%
\}
\newcommand{\glsxtrpostdescperson}{\%\ifgls\has\field{user1}{\glscurrententrylabel}\%
  \text{born}\%
  \ifgls\has\field{user2}{\glscurrententrylabel}\%
  \text{died}\%
  \glscurrentfieldvalue\%
  \%
  \%
  \%
\}%
\newcommand*{\glsxtrpostdescfilm}{% 
  \ifglshasfield{user1}{\glscurrententrylabel}{% 
    \glsxtrrestorepostpunc \ (released \glscurrentfieldvalue)}{% 
  }\ifglshasfield{user2}{\glscurrententrylabel}{% 
    \glsxtrrestorepostpunc \ featuring \glscurrentfieldvalue 
  }{% 
  }\glsxtrpostdescfilm

\renewcommand*{\glsxtrbookindexname}[1]{% 
  \glossentryname{#1}{% 
  \ifglshassymbol{#1}{% 
    \glsifcategory{#1}{chemical}{% 
      \glossentrysymbol{#1}{% 
        \space(\glossentrynameother{#1}{symbol})% 
      }% 
    }{\glsifcategory{#1}{film}{% 
      \ (film)% 
    }{% 
      }% 
    }% 
  }{% 
  }\glsxtrbookindexname

% requires glossaries-extra v1.25+:
\renewcommand*{\glsxtrpostnamehook}[1]{% 
  \glsadd[format=hyperbf]{#1}{% 
}\newcommand{\Unit}{\glsymbol}
\newcommand{\measurement}{\gls}
\glsxtrnewgls{film.}{\film}
\glsxtrnewglslike{idx.}{\idx}{\idxpl}{\Idx}{\Idxpl}

\begin{document}
\chapter{Sample}
\section{Bacteria}
This section is about \idxpl{bacteria}.
\subsection{First Use}
\gls{cbotulinum}, \gls{pputida}, \gls{cperfringens},
\gls{bsubtilis}, \gls{ctetani}, \gls{pcomposti},
\gls{pfimeticola}, \gls{cburnetii}, \gls{raustralis},
\gls{rrickettsii}.
\subsection{Next Use}
\gls{cbotulinum}, \gls{pputida}, \gls{cperfringens},
\gls{bsubtilis}, \gls{ctetani}, \gls{pcomposti},
\gls{pfimeticola}, \gls{cburnetii}, \gls{raustralis},
\gls{rrickettsii}.

\section{Markup Languages}
This section is about \idxpl{markuplanguage}.
\subsection{First Use}
\gls{LaTeX}, \gls{markdown}, \gls{xhtml}, \gls{mathml}, \gls{svg}.
\subsection{Next Use}
\gls{LaTeX}, \gls{markdown}, \gls{xhtml}, \gls{mathml}, \gls{svg}.

\section{Vegetables}
This section is about \idxpl{vegetable}.
\gls{cabbage}, \gls{brussels-sprout}, \gls{artichoke},
\gls{cauliflower}, \gls{courgette}, \gls{spinach}.

\section{Minerals}
This section is about \idxpl{mineral}.
\gls{beryl}, \gls{amethyst}, \gls{chalcedony}, \gls{aquamarine},
\gls{aragonite}, \gls{calcite}, \gls{bilinite},
\gls{cyanotrichite}, \gls{biotite}, \gls{dolomite},
\gls{quetzalcoatlite}, \gls{vulcanite}.

\section{Animals}
This section is about \idxpl{animal}.
\gls{duck}, \gls{parrot}, \gls{hedgehog}, \gls{sealioin},
\gls{zander}, \gls{aardvark}, \gls{zebra}, \gls{swan},
\gls{armadillo}.

\section{Chemicals}
This section is about \idxpl{chemical}.
\gls{Al2SO43} (\glsdesc{Al2SO43}), \gls{H2O} (\glsdesc{H2O}),
\gls{C6H12O6} (\glsdesc{C6H12O6}), \gls{CH3CH2OH}
(\glsdesc{CH3CH2OH}), \gls{CH2O} (\glsdesc{CH2O}), \gls{OF2}
(\glsdesc{OF2}), \gls{O2F2} (\glsdesc{O2F2}), \gls{SO42-}
(\glsdesc{SO42-}), \gls{H3O+} (\glsdesc{H3O+}), \gls{OH-}
8 Examples: sample-multi2.tex

\section*{SI Units}
\Idxpl{baseunit}: \Unit{ampere} (measures \measurement{ampere}), \Unit{kilogram} (measures \measurement{kilogram}), \Unit{metre}, \Unit{second}, \Unit{kelvin}, \Unit{mole}, \Unit{candela}.

\Idxpl{derivedunit}: \Unit{area}, \Unit{volume}, \Unit{acceleration}, \Unit{density}, \Unit{luminance}, \Unit{specificvolume}, \Unit{concentration}, \Unit{wavenumber}.

\section*{Books and Films}
\Idxpl{book}: \gls{ataleoftwocities} (by \gls{dickens}), \gls{thebigsleep} (by \gls{chandler}, \idx{film} adaptation: \film{thebigsleep}), \gls{icecoldinalex} (by \gls{landon}, \idx{film} adaptation: \film{icecoldinalex}), \gls{whydidnttheyaskevans} (by \gls{christie}, \idx{film} adaptation: \film{whydidnttheyaskevans}), \gls{doandroidsdreamofelectricsheep} (by \gls{dick}, inspired the \idx{film} \film{bladerunner}).

\Idxpl{film}: \film{anunexpectedjourney}, \film{desolationofsmaug} and \film{thebattleoffivearmies} (adapted from the \idx{book} \gls{thehobbit} by \gls{tolkien}), \film{thefellowshipofthering}, \film{thetwotowers} and \film{thereturnoftheking} (adapted from the \idx{book} \gls{thelordoftherings} also by \gls{tolkien}).

\section*{Miscellaneous Symbols}

\subsection*{First Use}
\Idxpl{mediacontrol}: \gls{forward}, \gls{forwardtoindex}, \gls{rewindtoindex}, \gls{rewind}.

\Idx{information}: \gls{bicycle}, \gls{coffeecup}, \gls{info}, \gls{gentsroom}, \gls{ladiesroom}, \gls{wheelchair}, \gls{football}, \gls{recycling}. 555
\Idx{weather}: \gls{cloud}, \gls{fog}, \gls{hail}, \gls{sun}, \gls{lightning}.

\subsection{Next Use}

\Idxpl{mediacontrol}: \gls{forward}, \gls{forwardtoindex}, \gls{rewindtoindex}, \gls{rewind}.

\Idx{information}: \gls{bicycle}, \gls{coffeecup}, \gls{info}, \gls{gentsroom}, \gls{ladiesroom}, \gls{wheelchair}, \gls{football}.

\Idx{weather}: \gls{cloud}, \gls{fog}, \gls{hail}, \gls{sun}, \gls{lightning}.

\section{Measurements}

\Idxpl{measurement}: \measurement{ampere}, \measurement{area}, \measurement{metre}.

\chapter{Glossaries}
\printunsrtglossary[type=bacteria,style=mcoltree]
\printunsrtglossary[type=markuplanguage,style=altlist]
\printunsrtglossary[type=vegetable,style=tree,nogroupskip]
\printunsrtglossary[type=mineral,style=treegroup]
\printunsrtglossary[type=animal,style=tree]
\printunsrtglossary[type=person,style=tree,nogroupskip]
\printunsrtglossary[type=book,style=tree,nogroupskip]
\printunsrtglossary[type=film,style=tree,nogroupskip]
\printunsrtglossary*[type=chemical,style=mcolalttreegroup]
{\%\renewcommand*{\glstreenamefmt}[1]{#1} %\renewcommand*{\glstreegroupheaderfmt}[1]{\textbf{#1}}%}
\printunsrtglossary*[type=measurement,style=alttree,nogroupskip]
{\%\renewcommand{\glsxtralttreeSymbolDescLocation}[2]{%\glossentrydesc{#1}%\ifglishassymbol{#1}{\space{\glossentrysymbol{#1}}}{}\glspostdescription\glsxtrAltTreePar}%}
\printunsrtglossary*[type=baseunit,style=alttree,nogroupskip]
{\%
\glsFindWidestTopLevelName[baseunit]\%
}
\printunsrtglossary[type=information,style=alttree,nogroupskip]
\printunsrtglossary[type=mediacontrol,style=alttree,nogroupskip]
\printunsrtglossary[type=weather,style=alttree,nogroupskip]
\printunsrtglossary*[type=index,style=bookindex]
{\
  \setupglossaries{section=chapter}\
  \let\sortname\swaptwo\
  \let\sortvonname\swapthree\
  \renewcommand*{\glsextrapostnamehook}{[1]{}}\
}
\end{document}
This section is about bacteria.

1.1.1 First Use
Clostridium botulinum, Pseudomonas putida, Bacillus subtilis, Rickettsia rickettsii, Rickettsia australis, Clostridium perfringens, Planifilum fimeticola.

1.9.2 Next Use
Media controls: ·, ¹, ´, ¶.
Information: ®, K, i, x, y, w, o.
Weather:  ,  ,  ,  ,  .

1.10 Measurements
Measurements: electric current, area, length.

1.1.2 Next Use
ettsia australis
8 Examples: sample-multi2.tex
Figure 8.23: sample-multi2.pdf (pages 9 and 12)
Package Option Summary

Most options are in the form ⟨option⟩=⟨value⟩ and may have a default if ⟨value⟩ is omitted, but some options don’t have values and should not have one assigned. For boolean options, if the value is omitted true is assumed. † Indicates a value that’s only provided by glossaries-extra and not by the base glossaries package.

A

abbreviations
  Creates the abbreviations glossary.
  Provided by glossaries-extra.

accsupp
  Load the glossaries-accsupp package to provide accessibility support.
  Provided by glossaries and modified by glossaries-extra.

acronym=⟨boolean⟩
  If true, creates a new glossary with the label acronym.
  Provided by glossaries.

acronymlists=⟨list⟩
  Identifies the glossaries that are lists of acronyms (don’t use with glossaries-extra).
  Provided by glossaries.

acronyms
  Equivalent to acronym={true}.
  Provided by glossaries.

automake=⟨boolean⟩
  If true, tries to use \TeX’s shell escape to automatically run the required indexing application (may not be permitted by \TeX’s security settings).
  Provided by glossaries.

autoseeindex=⟨boolean⟩
  If true, the see and seealso keys automatically indexes the cross-referenced term.
  Provided by glossaries-extra. Not relevant with bib2gls.

B
counter=⟨value⟩
  Sets the default location counter to ⟨value⟩ (which must be the name of a counter). May be overridden on an individual basis using the counter option in commands like \gls and \glsadd.
  Provided by glossaries.

counterwithin=⟨counter name⟩
  Automatically sets the option entrycounter={true} and indicates the master counter for glossaryentry.
  Provided by glossaries.

ddebug=⟨value⟩
  Add debugging information; allowed values: false (default), true (info added to transcript), showtargets (info added to transcript and show target name in the document for glossary-related hyperlinks), showwrgloss† show mark in document where indexing occurs and all† (implement both showtargets and showwrgloss).
  Provided by glossaries and modified by glossaries-extra.

561
docdef=⟨value⟩
Determines whether entries can be defined in the document environment; the ⟨value⟩ may be one of: false (entries must be defined in the preamble), true (entries may be defined in the document environment), restricted (entries may only be defined in the document environment if the definition comes before all glossaries and before any reference to the entry).
Provided by glossaries-extra. Not relevant with bib2gls.

entrycounter=⟨boolean⟩
If true, creates the glossaryentry counter and each main (level 0) glossary entry will be numbered (which can be referenced with \glsrefentry{} or \glsxtrpageref{}).
Provided by glossaries.

esclocations=⟨boolean⟩
If true, glossaries tries to escape special characters from the locations.
Provided by glossaries. Not relevant with bib2gls.

hyperfirst=⟨boolean⟩
If false, terms on first use don’t have hyperlinks unless explicitly set (with glossaries-extra, the nohyperfirst category attribute can selectively apply this).
Provided by glossaries.

index
Defines the index glossary and \newterm. Provided by glossaries.

indexcounter
Creates the wrglossary counter, which is incremented every time an entry is indexed with that counter, and sets that as the default location counter.
Provided by glossaries-extra.

indexcrossrefs=⟨boolean⟩
If true, at the end of the document automatically index cross-referenced entries that haven’t been marked as used.
Provided by glossaries-extra. Not relevant with bib2gls.

indexonlyfirst=⟨boolean⟩
If true, only performs indexing on first use.
Provided by glossaries.

makeindex
Write the indexing information using makeindex’s format.
Provided by glossaries. Not relevant with bib2gls.

nogroupskip=⟨boolean⟩
If true, suppresses the visual separation between letter groups in glossary styles that support this option.
Provided by glossaries.

nohypertypes=⟨list⟩
Suppress hyperlinks for the listed glossary types.
Provided by glossaries.

nolangwarn
 Suppresses warnings generated by missing language modules.
Provided by glossaries.

nolist
Prevents the glossary-list package (which provides the list styles) from being automatically loaded.
Provided by glossaries.
nolong
Prevents the glossary-long package (which provides the long styles) from being automatically loaded.
Provided by glossaries.

nomain
Suppresses the creation of the default main glossary. If used an alternative glossary must be created.
Provided by glossaries.

nomissingglistext=⟨boolean⟩
If true, suppresses the warning text that appears in the document with \printglossary if the associated external file doesn’t exist.
Provided by glossaries-extra. Not relevant with bib2gls.

nonumberlist
Suppresses the location lists from being displayed in the glossary lists (the package option isn’t boolean, but the option of the same name for \printglossary, \printunsrtglossary and \printnoidxglossary is boolean); with bib2gls you can use save-locations={false} instead.
Provided by glossaries.

nopostdot=⟨boolean⟩
If true, suppresses the automatic post-description punctuation. With glossaries-extra you can also use postpunc={none} instead of nopostdot={true} and postdot or postpunc={dot} instead of nopostdot={false}.
Provided by glossaries and modified by glossaries-extra.

noredefwarn
Suppresses warnings if overriding glossary commands provided by another class or package.
Provided by glossaries.

nostyles
Prevents all the default styles from being loaded. If this option is used a style must be defined in the document or a package providing a style needs to be loaded (either through stylemods or with \usepackage).
Provided by glossaries.

nosuper
Prevents the glossary-super package (which provides the super styles) from being automatically loaded.
Provided by glossaries.

notranslate
Equivalent to translate={false}.
Provided by glossaries.

notree
Prevents the glossary-tree package (which provides the tree styles) from being automatically loaded.
Provided by glossaries.

nowarn
Suppresses all glossaries-related warnings.
Provided by glossaries.

numberedsection=⟨value⟩
Determines whether to use numbered or unnumbered section units, and whether or not to automatically add \label; the value may be one of: false (default, no numbering and no label), no label (numbered but no label), autolabel (numbered with automatic label), nameref (unnumbered but labelled). If no value is given no label is assumed.
Provided by glossaries.

numberline=⟨boolean⟩
When used with toc={true}, this will add \numberline{} to the start of the TOC entry.
Provided by glossaries.

numbers
Defines the numbers glossary; with glossaries-extra additionally defines \glstextrnewnumber.
Provided by glossaries and modified by glossaries-extra.
Package Option Summary

O

order=(value)
Sets whether to use word or letter ordering.
Provided by glossaries. Not relevant with bib2gls, use the break-at resource option instead.

P

postdot
Equivalent to postpunc={dot}.
Provided by glossaries-extra.

postpunc=(value)
Controls the automatic post-description punctuation; the value may be one of: none (not required, the description or glossary style already supplies the terminating punctuation), comma (use a comma), dot (use a full stop with the space factor adjusted), (punctuation) (use ⟨punctuation⟩).
Provided by glossaries-extra.

R

record=(value)
Unless the value is off, this option sets up glossaries-extra for use with bib2gls: only (assumed if no (value) supplied) indexing is performed by bib2gls; nameref (glossaries-extra v1.37+) like only but includes extra information in the records; alsoindex (hybrid method) bib2gls is used to provide the entry definitions but makeindex or xindy is used for the indexing.
Provided by glossaries-extra.

S

sanitizesort=(boolean)
Determines whether or not to sanitize the sort key (not relevant with bib2gls).
Provided by glossaries. Not relevant with bib2gls.

savenumberlist=(boolean)
If true, stores the number list for each entry (with bib2gls use the save-locations resource option instead).
Provided by glossaries.

savewrites=(boolean)
If true, indexing information is stored in token registers that are only written at the end of the document to save creating a write register per glossary indexing file.
Provided by glossaries. Not relevant with bib2gls.

section=(value)
Indicates the sectional unit to use for the glossary heading (the value should be the name of the section command without the leading backslash, for example section={subsection}). If no value is supplied then section={section} is assumed. If this option is omitted, then the default is either section={chapter} or section={section}, depending on whether or not \chapter has been defined. The starred or unstarred version is determined by numberedsection.
Provided by glossaries.

seeautonumberlist
If nonumberlist is used, this allows the see key to override the setting for the associated entry.
Provided by glossaries. Not relevant with bib2gls.

seenoindex=(value)
Determines whether the see key automatically indexes the entry using \glsadd; allowed values: error (attempts indexing but triggers an error if used before \makeglossaries); warn (attempts indexing but triggers a warning if used before \makeglossaries); ignore (attempts indexing but does nothing if used before \makeglossaries).
Provided by glossaries. Not relevant with bib2gls.
Package Option Summary

**shortcuts=⟨value⟩**

Sets up short cut commands; the value may be one of false (default), true (assumed if no value supplied), implements shortcuts = {ac}, shortcuts = {abbreviations} and shortcuts = {other}, acronyms† (equivalent to base shortcuts = {true}, synonym acro), ac† (provides \ac shortcuts that use glossaries-extra’s new abbreviation commands), abbreviations† (provides \ab shortcuts), other† (provides other shortcut commands), all† (synonym for shortcuts = {true}) and none† (synonym for shortcuts = {false}).

Provided by glossaries.

**sort=⟨value⟩**

Indicates how to assign the sort key if not explicitly set, the value may be one of: none (don’t automatically assign the sort field), standard (obtain the sort value from the name field), def (assign the sort field to a numerical value that represents the order of definition), user (assign the sort field to a numerical value that represents the order of first use).

Provided by glossaries. Not relevant with bib2gls, use the sort resource option instead.

**style=⟨name⟩**

Sets the default glossary style to ⟨name⟩.

Provided by glossaries.

**stylemods=⟨value⟩**

Load the glossaries-extra–stylemods package with the supplied options (which should be a list of suffix parts identifying glossary style packages glossary-⟨suffix⟩); there are two keyword values: default (equivalent to omitting ⟨value⟩) and all, which loads all predefined styles.

Provided by glossaries-extra.

**subentrycounter=⟨boolean⟩**

If true, creates the glossary subentry counter and each level 1 glossary entry will be numbered (which can be referenced with \glsrefentry or \glsxtrpageref); this option and associated counter are independent of entrycounter and glossaryentry.

Provided by glossaries.

**symbols**

Defines the symbols glossary; with glossaries-extra additionally defines \glsxtrnewsymbol.

Provided by glossaries and modified by glossaries-extra.

**toc=⟨boolean⟩**

If true (default for glossaries-extra), automatically add each glossary to the table of contents.

Provided by glossaries.

**translate=⟨value⟩**

Determines the multilingual support provided by glossaries; allowed values: true (default with just base glossaries; if babel has been loaded and translator is installed, use translator interface), false (don’t provide translations), babel (default with glossaries-extra; don’t load the translator package, just load glossaries-babel).

Provided by glossaries.

**ucmark=⟨boolean⟩**

If true, converts the glossary mark (used in page headings) to upper case with \MakeTextUppercase.

Provided by glossaries.

**undefaction=⟨value⟩**

Indicates what to do if an undefined entry is referenced: warn (generate a warning and
show ?? in the text, default with record), error (generate an error).

Provided by glossaries-extra.

xindy=⟨settings⟩

Write the indexing information using xindy’s format where the optional ⟨settings⟩ may supply the language and code page and whether or not to define the default number group.

Provided by glossaries. Not relevant with bib2gls.

xindygloss

Equivalent to xindy={}. 

Provided by glossaries. Not relevant with bib2gls.

xindynoglsnumbers

Equivalent to xindy={glnumbers=false}. 

Provided by glossaries. Not relevant with bib2gls.
General Command Summary

This is an alphabetical summary of commands referenced in this document. See the relevant user guides for further details.

\} indicates command is recognised by bib2gls’s interpreter although it may have a slightly different implementation.

\{ kernel command*
Produces the open brace symbol {.

\{ kernel command*
Produces the close brace symbol }.

" bib2gls quark
A quark that identifies a literal double-quote ".

"{⟨character⟩} kernel command*
Puts an umlaut accent over ⟨character⟩.

\# kernel command*
Produces the hash symbol #.

\# bib2gls quark
A quark that identifies a literal hash # in a regular expression.

\$ kernel command*
Produces the dollar symbol $.

\$ bib2gls quark
A quark that identifies a literal dollar $ in a regular expression.

\% kernel command*
Produces the percent symbol %.

\& kernel command*
Produces the ampersand symbol &.

\& bib2gls quark
A quark that identifies a literal ampersand & in a regular expression.

\\{⟨character⟩} kernel command*
Puts an acute accent over ⟨character⟩.

\\ bib2gls quark
A quark that identifies a literal open parenthesis ( in a regular expression.
General Command Summary

\)  
A quark that identifies a literal close parenthesis \) in a regular expression.

\*  
A quark that identifies a literal star * in a regular expression.

\+  
A quark that identifies a literal plus + in a regular expression.

\,  
Thin space.

\-  
A quark that identifies a literal hyphen – in a regular expression.

\{⟨character⟩\}  
Puts a dot accent over ⟨character⟩.

\.  
A quark that identifies a literal dot (.) in a regular expression.

\/  
A quark that identifies a literal slash / in a regular expression.

\:  
A quark that identifies a literal colon : in a regular expression.

\<  
A quark that identifies a literal less than < in a regular expression.

\>  
A quark that identifies a literal greater than > in a regular expression.

\?  
A quark that identifies a literal question mark ? in a regular expression.

\@  
Adjusts the space factor to indicate the following punctuation character marks the end of the sentence.

\@currentHref  
Used to store the current anchor for the next instance of \label.

\@currentlabelname  
Used to store the current title information for the next instance of \label.

\@firstofone{⟨code⟩}  
Does ⟨code⟩.

\@for(⟨cs⟩):=(⟨list⟩)\do{⟨code⟩}  
Iterates over each item in the comma-separated ⟨list⟩, and on each iteration sets ⟨cs⟩ to the current element and performs ⟨code⟩.
General Command Summary

\@gls@hypergroup\{\langle type\rangle\}{\langle group id\rangle}\}
glossary-hypernav
Command written to the .aux file that identifies that the given group was used in the
glossary on the previous run.

\@gobble\{\langle code\rangle\}\}
kernal command*
Does nothing (the argument is discarded).

\@istfilename\{\langle filename\rangle\}\}
glossaries
Identifies the style file in the .aux file for the benefit of external tools like
makeglossaries and makeglossaries-lite.

\[
A quark that identifies a literal open square bracket [ in a regular expression.
\\[
\langle\langle len\rangle\rangle\]
kernal command*
Starts a new row in a tabular or array context with an extra vertical space of length \langle len\rangle
above it (starred form prohibits a page break).
\]\bib2gls quark
A quark that identifies a literal backslash \ in a regular expression.
\]\bib2gls quark
A quark that identifies a literal close square bracket ] in a regular expression.
\~\{\langle character\rangle\}\}
kernal command*
Puts a circumflex accent over \langle character\rangle.
\^bib2gls quark
A quark that identifies a literal circumflex ^ in a regular expression.
\_bib2gls quark
Produces the underscore symbol _.
\|bib2gls quark
A quark that identifies a literal pipe character | in a regular expression.
\~bib2gls quark
A quark that identifies a literal tilde ~ in a regular expression.
\~bib2gls quark
Produces an inter-word space.

\AA
Produces the upper case A-ring character Å.
\aa
Produces the lower case a-ring character å.
\ab\{\langle options\rangle\}{\langle label\rangle}\}{\langle insert\rangle}\}
glossaries-extra shortcuts={abbreviations}
Equivalent to \gls.
\abbreviationname \glossaries-extra
Language-sensitive name used for the title of the glossary created with the \abbreviations package option.

\abbrvpluralsuffix \glossaries-extra
The style sensitive suffix used to construct the default plural for the short form of abbreviations.

\ac[\langle options\rangle][\langle label\rangle][\langle insert\rangle] \glossaries-extra \shortcuts
Equivalent to \gls.

\acronymname \glossaries
Language-sensitive name used for the title of the glossary created with the \acronym or \acronyms package option.

\acronymtype \glossaries
Expands to the default acronym glossary type when using \newacronym.

\acrpluralsuffix \glossaries
The suffix used to construct the default plural for the short form of acronyms using the base \glossaries package’s acronym mechanism (not used with the \glossaries-extra enhanced abbreviation mechanism).

\AE \kernelcommand
Produces the upper case Æ-ligature.

\ae \kernelcommand
Produces the lower case æ-ligature.

\alph\{\langle counter\rangle\} \kernelcommand
Displays the given counter as an alphabetic character from “a” to “z”.

\Alpha \glossaries-extra-bib2gls
Greek letter alpha A.

\alpha \kernelcommand (maths mode)
Greek letter alpha \(\alpha\).

\alsoname \ languagelanguages
Language sensitive “see also” text.

\approx \kernelcommand (maths mode)
Approximate symbol \(\approx\).

\appto\{\langle cs\rangle\}\{\langle code\rangle\} \etoolbox
Appends \langle code\rangle to the definition of the control sequence \langle cs\rangle.

\apptoglossarypreamble[\langle type\rangle]\{\langle code\rangle\} \glossaries-extra
Appends \langle code\rangle to the preamble for the given glossary (or the default of \langle type\rangle is omitted).

\AtEndDocument\{\langle code\rangle\} \kernelcommand
Perform \langle code\rangle at the end of the document.
General Command Summary

\autoref{⟨id⟩}
Cross-reference with textual tag inferred from the associated counter.

B

\backmatter
Switches to back matter.

\bfseries
Switch to bold (until end of current scope).

\bibglsaliassep
Separator between alias cross-reference and location list.

\bibglsampersandchar
Expands to a literal ampersand character.

\bibglsaposchar
Expands to a single-quote (‘) character.

\bibglscircumchar
Expands to a literal circumflex character.

\bibglscompact{⟨pattern⟩}{⟨part1⟩}{⟨part2⟩}
Compaction used on the end range location.

\bibglscontributor{⟨forenames⟩}{⟨von-part⟩}{⟨surname⟩}{⟨suffix⟩}
Used to markup a contributor’s name that was converted from \TeX ‘s contributor syntax.

\bibglscontributorlist{⟨list⟩}{⟨number⟩}
Used to markup a list of names from a field that was converted from \TeX ‘s contributor syntax.

\bibglscopytoglossary{⟨entry-label⟩}{⟨glossary-type⟩}
Copies the entry identified by ⟨entry-label⟩ to the glossary identified by ⟨glossary-type⟩ after the entry is defined (if the copy-to-glossary option is set).

\bibglsdate{⟨year⟩}{⟨month⟩}{⟨day-of-month⟩}{⟨day-of-week⟩}{⟨day-of-year⟩}{⟨era⟩}
{⟨original⟩}
Used to markup a date converted from a field value.

\bibglsdategroup{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}{⟨G⟩}{⟨title⟩}{⟨group-id⟩}{⟨type⟩}
Expands to the date group label.

\bibglsdategrouphier{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}{⟨G⟩}{⟨title⟩}{⟨group-id⟩}{⟨type⟩}{⟨parent⟩}
{⟨level⟩}
Expands to the hierarchical date group label.

\bibglsdategrouptitle{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}{⟨G⟩}{⟨title⟩}{⟨group-id⟩}{⟨type⟩}
Expands to the date group title.
General Command Summary

\bibglsdategrouptitlehier\{YYYY\}{(MM)}{(DD)}{(G)}{(title)}{(group-id)}{(type)}
\{parent\}{(level)}

Expands to the hierarchical date group title.

\bibglsdatetimeremainder\{millisec\}{(dst)}{(zone)}{(original)}

Used internally to pick up the final four arguments of \bibglsdatetimethe
time.

\bibglssdefcompoundset\{options\}{(label)}{(main)}{(elements)}

Defines a compound entry provided with \@compoundset.

\bibglsdefinitionindex\{label\}

Expands to the definition index of the entry identified \{label\} if
save-definition-index is set otherwise expands to empty.

\bibglsdelimN

Separator for individual locations (except last).

\bibglsdollarchar

Expands to a literal dollar character.

\bibglsdoublequotechar

Expands to a double-quote (" ) character.

\bibglsdualprefixlabel\{prefix\}

Hook provided to pick up the dual prefix, if required.
\bibglsemptygroup{⟨type⟩}
Expands to the empty group label.

\bibglsemptygrouphier{⟨type⟩}{⟨parent⟩}{⟨level⟩}
Expands to the hierarchical empty group label.

\bibglsemptygrouptitle{⟨type⟩}
Expands to the empty group title.

\bibglsemptygrouptitlehier{⟨type⟩}{⟨parent⟩}{⟨level⟩}
Expands to the hierarchical empty group title.

\bibglsexternalprefixlabel{⟨n⟩}{⟨prefix⟩}
Hook provided to pick up the ⟨n⟩th external prefix, if required.

\bibglsfirstuc{⟨text⟩}
Converts the first letter of ⟨text⟩ to upper case.

\bibglsflattenedchildpostsort{⟨parent name⟩}{⟨child name⟩}
Expands to the post-sort flattened child entry’s new name.

\bibglsflattenedchildpresort{⟨child name⟩}{⟨parent name⟩}
Expands to the pre-sort flattened child entry’s new name.

\bibglsflattenedhomograph{⟨name⟩}{⟨parent label⟩}
Expands to the flattened entry’s new name.

\bibglsgrouplevel{⟨label⟩}{⟨n⟩}
Expands to the sub-group label for hierarchical level ⟨n⟩ where ⟨label⟩ is the label that would normally be applied to level 0.

\bibglshashchar
Expands to a literal hash character #.

\bibglshexunicodechar{⟨hex⟩}
Produces the character with the given hexadecimal code.

\bibglshiersubgrouptitle{⟨level⟩}{⟨parent⟩}{⟨title⟩}
Formats the title for a hierarchical group where the normal non-hierarchical title would be ⟨title⟩.

\bibglshrefchar{⟨hex⟩}{⟨char⟩}
Expands to a literal percent character followed by ⟨hex⟩.

\bibglshrefunicode{⟨hex⟩}{⟨char⟩}
Expands to ⟨char⟩ by default.

\bibglshypergroup{⟨type⟩}{⟨group-id⟩}
Creates group navigation information.

\bibglshyperlink{⟨text⟩}{⟨label⟩}
Displays ⟨text⟩ with a hyperlink to the entry given by ⟨label⟩, if supported.

\bibglsinterloper{⟨location⟩}
Interloper location format.
General Command Summary

\texttt{\textbackslash bibglslastDelimN}
Separator before the last location (where there is more than one location).

\texttt{\textbackslash bibglslettergroup}\{\texttt{\langle title\rangle}\{\texttt{\langle letter\rangle}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}}
Expands to the letter group label.

\texttt{\textbackslash bibglslettergrouphier}\{\texttt{\langle title\rangle}\{\texttt{\langle letter\rangle}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}{\texttt{\langle parent\rangle}\}{\texttt{\langle level\rangle}\}}
Expands to the hierarchical letter group label.

\texttt{\textbackslash bibglslettergrouptitle}\{\texttt{\langle title\rangle}\{\texttt{\langle letter\rangle}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}}
Expands to the letter group title.

\texttt{\textbackslash bibglslettergrouptitlehier}\{\texttt{\langle title\rangle}\{\texttt{\langle letter\rangle}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}{\texttt{\langle parent\rangle}\}{\texttt{\langle level\rangle}\}}
Expands to the hierarchical letter group title.

\texttt{\textbackslash bibglslocationgroup}\{\texttt{\langle n\rangle}\{\texttt{\langle counter\rangle}\}{\texttt{\langle list\rangle}\}}
Location group encapsulator.

\texttt{\textbackslash bibglslocationgrouppsep}
Location group separator.

\texttt{\textbackslash bibglslocprefix}\{\texttt{\langle n\rangle}\}
Location list prefix.

\texttt{\textbackslash bibglslocsuffix}\{\texttt{\langle n\rangle}\}
Location list suffix.

\texttt{\textbackslash bibglslowercase}\{\texttt{\langle text\rangle}\}
Converts \texttt{\langle text\rangle} to lower case.

\texttt{\textbackslash bibglsmergedgroup}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}{\texttt{\langle n\rangle}\}{\texttt{\langle g_1\rangle}\{\texttt{\langle g_2\rangle}...{\texttt{\langle g_{n-1}\rangle}}\{\texttt{\langle g_n\rangle}\}}
Expands to the merged group label.

\texttt{\textbackslash bibglsmergedgrouppfmt}\{\texttt{\langle n\rangle}\}{\texttt{\langle g_1\rangle}\{\texttt{\langle g_2\rangle}...{\texttt{\langle g_{n-1}\rangle}}\{\texttt{\langle g_n\rangle}\}}
Used by \texttt{\textbackslash bibglsmergedgrouptitle} and \texttt{\textbackslash bibglsmergedgrouphierfmt} to format the title.

\texttt{\textbackslash bibglsmergedgrouphier}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}{\texttt{\langle n\rangle}\}{\texttt{\langle g_1\rangle}\{\texttt{\langle g_2\rangle}...{\texttt{\langle g_{n-1}\rangle}}\{\texttt{\langle g_n\rangle}\}{\texttt{\langle parent\rangle}\}{\texttt{\langle level\rangle}\}}
Expands to the merged hierarchical group label.

\texttt{\textbackslash bibglsmergedgrouphierfmt}\{\texttt{\langle n\rangle}\}{\texttt{\langle g_1\rangle}\{\texttt{\langle g_2\rangle}...{\texttt{\langle g_{n-1}\rangle}}\{\texttt{\langle g_n\rangle}\}}
Used by \texttt{\textbackslash bibglsmergedgrouptitlehier} to format the title.

\texttt{\textbackslash bibglsmergedgrouptitle}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}{\texttt{\langle n\rangle}\}{\texttt{\langle g_1\rangle}\{\texttt{\langle g_2\rangle}...{\texttt{\langle g_{n-1}\rangle}}\{\texttt{\langle g_n\rangle}\}}
Expands to the merged group title.

\texttt{\textbackslash bibglsmergedgrouptitlehier}\{\texttt{\langle id\rangle}\{\texttt{\langle type\rangle}\}{\texttt{\langle n\rangle}\}{\texttt{\langle g_1\rangle}\{\texttt{\langle g_2\rangle}...{\texttt{\langle g_{n-1}\rangle}}\{\texttt{\langle g_n\rangle}\}{\texttt{\langle parent\rangle}\}{\texttt{\langle level\rangle}\}}
Expands to the merged hierarchical group title.

\texttt{\textbackslash bibglssnewabbreviation}\{\texttt{\langle label\rangle}\}{\texttt{\langle options\rangle}\}{\texttt{\langle short\rangle}\}{\texttt{\langle long\rangle}\}
Defines terms provided with \texttt{\@abbreviation}.
General Command Summary

\bibglsnewacronym{\{label\}}{\{options\}}{\{short\}}{\{long\}}\hfill bib2gls
Defines terms provided with @acronym.

\bibglsnewbibtexentry{\{label\}}{\{options\}}{\{name\}}{\{description\}}\hfill bib2gls
Defines terms provided with @bibtexentry.

\bibglsnewcontributor{\{label\}}{\{options\}}{\{name\}}{\{description\}}\hfill bib2gls
Defines terms provided with @contributor.

\bibglsnewdualabbreviation{\{label\}}{\{options\}}{\{short\}}{\{long\}}\hfill bib2gls
Defines terms provided with @dualabbreviation.

\bibglsnewdualabbreviationentry{\{label\}}{\{options\}}{\{short\}}{\{long\}} \{\{description\}\}\hfill bib2gls
Defines primary terms provided with @dualabbreviationentry.

\bibglsnewdualabbreviationentrysecondary{\{label\}}{\{options\}}{\{short\}}{\{long\}} \{\{description\}\}\hfill bib2gls
Defines secondary terms provided with @dualabbreviationentry.

\bibglsnewdualacronym{\{label\}}{\{options\}}{\{short\}}{\{long\}}\hfill bib2gls
Defines terms provided with @dualacronym.

\bibglsnewdualentry{\{label\}}{\{options\}}{\{name\}}{\{description\}}\hfill bib2gls
Defines terms provided with @dualentry.

\bibglsnewdualentryabbreviation{\{label\}}{\{options\}}{\{short\}}{\{long\}} \{\{description\}\}\hfill bib2gls
Defines primary terms provided with (deprecated) @dualentryabbreviation.

\bibglsnewdualentryabbreviationsecondary{\{label\}}{\{options\}}{\{short\}}{\{long\}} \{\{description\}\}\hfill bib2gls
Defines secondary terms provided with (deprecated) @dualentryabbreviation.

\bibglsnewdualindexabbreviation{\{label\}}{\{dual-label\}}{\{options\}}{\{name\}}{\{short\}} \{\{long\}}{\{description\}\}\hfill bib2gls
Defines primary terms provided with @dualindexabbreviation.

\bibglsnewdualindexabbreviationsecondary{\{label\}}{\{options\}}{\{name\}}{\{short\}} \{\{long\}}{\{description\}\}\hfill bib2gls
Defines secondary terms provided with @dualindexabbreviation.

\bibglsnewdualindexentry{\{label\}}{\{options\}}{\{name\}}{\{description\}}\hfill bib2gls
Defines primary terms provided with @dualindexentry.

\bibglsnewdualindexentrysecondary{\{label\}}{\{options\}}{\{name\}}{\{description\}}\hfill bib2gls
Defines secondary terms provided with @dualindexentry.

\bibglsnewdualindexnumber{\{label\}}{\{options\}}{\{name\}}{\{symbol\}}{\{description\}}\hfill bib2gls
Defines primary terms provided with @dualindexnumber.

\bibglsnewdualindexnumbersecondary{\{label\}}{\{options\}}{\{name\}}{\{description\}}\hfill bib2gls
Defines secondary terms provided with @dualindexnumber.

575
General Command Summary

\bibglsnewdualindexsymbol{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle symbol \rangle}{\langle description \rangle}
\bib2gls
Defines primary terms provided with \dualindexsymbol.

\bibglsnewdualindexsymbolsecondary{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle description \rangle}
\bib2gls
Defines secondary terms provided with \dualindexsymbol.

\bibglsnewdualnumber{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle description \rangle}
\bib2gls
Defines terms provided with \dualnumber.

\bibglsnewdualsymbol{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle description \rangle}
\bib2gls
Defines terms provided with \dualsymbol.

\bibglsnewentry{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle description \rangle}
\bib2gls
Defines terms provided with \entry.

\bibglsnewindex{\langle label \rangle}{\langle options \rangle}
\bib2gls
Defines terms provided with \index.

\bibglsnewindexplural{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}
\bib2gls
Defines terms provided with \indexplural.

\bibglsnewnumber{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle description \rangle}
\bib2gls
Defines terms provided with \number.

\bibglsnewprogenitor{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle description \rangle}
\bib2gls
Defines terms provided with \progenitor.

\bibglsnewspawnabbreviation{\langle label \rangle}{\langle options \rangle}{\langle short \rangle}{\langle long \rangle}
\bib2gls
Defines terms provided with \spawnabbreviation.

\bibglsnewspawnacronym{\langle label \rangle}{\langle options \rangle}{\langle short \rangle}{\langle long \rangle}
\bib2gls
Defines terms provided with \spawnacronym.

\bibglsnewspawndualindexentry{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}{\langle description \rangle}
\bib2gls
Defines terms provided with \spawndualindexentry.

\bibglsnewspawndualindexentrysecondary{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}
\bib2gls
Defines secondary terms provided with \spawndualindexentry.

\bibglsnewspawnedabbreviation{\langle label \rangle}{\langle options \rangle}{\langle short \rangle}{\langle long \rangle}
\bib2gls
Defines terms spawned from \spawnabbreviation.

\bibglsnewspawnedacronym{\langle label \rangle}{\langle options \rangle}{\langle short \rangle}{\langle long \rangle}
\bib2gls
Defines terms spawned from \spawnacronym.

\bibglsnewspawnedentry{\langle label \rangle}{\langle options \rangle}
\bib2gls
Defines terms spawned from \spawnentry.

\bibglsnewspawnedindex{\langle label \rangle}{\langle options \rangle}
\bib2gls
Defines terms spawned from \spawnindex or \progenitor.

\bibglsnewspawnedindexplural{\langle label \rangle}{\langle options \rangle}{\langle name \rangle}
\bib2gls
Defines terms spawned from \spawnindexplural.
General Command Summary

\bibglsnewspawnednumber\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms spawned from @spawnnumber.

\bibglsnewspawnedsymbol\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms spawned from @spawnsymbol.

\bibglsnewspawnentry\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms provided with @spawnentry.

\bibglsnewspawnindex\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms provided with @spawnindex.

\bibglsnewspawnindexplural\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms provided with @spawnindexplural.

\bibglsnewspawnnumber\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms provided with @spawnnumber.

\bibglsnewspawnsymbol\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms provided with @spawnsymbol.

\bibglsnewsymbol\{⟨label⟩\}{⟨options⟩}{⟨name⟩}{⟨description⟩}
Defines terms provided with @symbol.

\bibglsnewtertiaryindexabbreviationentry\{⟨label⟩\}{⟨dual-label⟩}{⟨options⟩}{⟨name⟩}{⟨short⟩}{⟨long⟩}{⟨description⟩}
Defines primary terms provided with @tertiaryindexabbreviationentry.

\bibglsnewtertiaryindexabbreviationentrysecondary\{⟨label⟩\}{⟨tertiary-label⟩}{⟨options⟩}{⟨tertiary-opts⟩}{⟨primary-name⟩}{⟨short⟩}{⟨long⟩}{⟨description⟩}
Defines secondary and tertiary terms provided with @tertiaryindexabbreviationentry.

\bibglsnumbergroup\{⟨value⟩\}{⟨id⟩}{⟨type⟩}
Expands to the number group label.

\bibglsnumbergrouphier\{⟨value⟩\}{⟨id⟩}{⟨type⟩}{⟨parent⟩}{⟨level⟩}
Expands to the hierarchical number group label.

\bibglsnumbergroup/title\{⟨value⟩\}{⟨id⟩}{⟨type⟩}
Expands to the number group title.

\bibglsnumbergrouptitlehier\{⟨value⟩\}{⟨id⟩}{⟨type⟩}{⟨parent⟩}{⟨level⟩}
Expands to the hierarchical number group title.

\bibglsothergroup\{⟨character⟩\}{⟨id⟩}{⟨type⟩}
Expands to the non-letter group label.

\bibglsothergrouphier\{⟨character⟩\}{⟨id⟩}{⟨type⟩}{⟨parent⟩}{⟨level⟩}
Expands to the hierarchical non-letter group label.

\bibglsothergrouptitle\{⟨character⟩\}{⟨id⟩}{⟨type⟩}
Expands to the non-letter group title.
General Command Summary

\bibglsothergrouptitlehier{{\texttt{character}}}{{\texttt{id}}}{{\texttt{type}}}{{\texttt{parent}}}{{\texttt{level}}} \quad \text{bib2gls}
Expands to the hierarchical non-letter group title.

\bibglspagename \quad \text{bib2gls}
Name used for single page.

\bibglspagesname \quad \text{bib2gls}
Name used for multiple pages.

\bibglspassim \quad \text{bib2gls}
Passim range suffix.

\bibglspassimname \quad \text{bib2gls}
Name used by passim range suffix.

\bibglspostlocprefix \quad \text{bib2gls}
Location list post prefix.

\bibglprimary{n}{{\texttt{locations}}} \quad \text{bib2gls}
Location list encapsulator used in the primarylocations field.

\bibglprimarylocationgroup{n}{{\texttt{counter}}}{{\texttt{list}}} \quad \text{bib2gls}
Primary location group encapsulator.

\bibglprimarylocationgroupsep \quad \text{bib2gls}
Primary location group separator.

\bibglprimaryprefixlabel{{\texttt{prefix}}} \quad \text{bib2gls}
Hook provided to pick up the primary prefix, if required.

\bibglsrangle{\texttt{start}}\delimR{\texttt{end}} \quad \text{bib2gls}
Explicit range format.

\bibglssseealsosep \quad \text{bib2gls}
Separator between seealso cross-references and location list.

\bibglssseesep \quad \text{bib2gls}
Separator between see cross-references and location list.

\bibglsssetdatetitlegrouptitle{{{\texttt{YYYY}}}{{{\texttt{MM}}}{{{\texttt{DD}}}{{{\texttt{G}}}{{{\texttt{title}}}{{{\texttt{group-id}}}{{{\texttt{type}}}}}})}}} \quad \text{bib2gls}
Sets the date (no time) group title.

\bibglsssetdatetitlegrouphier{{{\texttt{YYYY}}}{{{\texttt{MM}}}{{{\texttt{DD}}}{{{\texttt{G}}}{{{\texttt{title}}}{{{\texttt{group-id}}}{{{\texttt{type}}}{{{\texttt{parent}}}{{{\texttt{level}}}}}}}}}}}} \quad \text{bib2gls}
Sets the hierarchical date (no time) group title.

\bibglsssetdatetimegrouptitle{{{\texttt{YYYY}}}{{{\texttt{MM}}}{{{\texttt{DD}}}{{{\texttt{hh}}}{{{\texttt{mm}}}{{{\texttt{ss}}}{{{\texttt{zone}}}}}}}}}{{{\texttt{title}}}{{{\texttt{group-id}}}{{{\texttt{type}}}}}} \quad \text{bib2gls}
Sets the date-time group title.

\bibglsssetdatetimegrouphier{{{\texttt{YYYY}}}{{{\texttt{MM}}}{{{\texttt{DD}}}{{{\texttt{hh}}}{{{\texttt{mm}}}{{{\texttt{ss}}}{{{\texttt{zone}}}}}}}}}{{{\texttt{title}}}{{{\texttt{group-id}}}{{{\texttt{type}}}{{{\texttt{parent}}}{{{\texttt{level}}}}}}}} \quad \text{bib2gls}
Sets the hierarchical date-time group title.
\bibglssetemptygrouptitle{\{(type)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the empty group title.

\bibglssetemptygrouptitlehier{\{(type)\}\{(parent)\}\{(level)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the hierarchical empty group title.

\bibglssetlastgrouptitle{\{cs\}\{(specs)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the last group title.

\bibglssetlettergrouptitle{\{(title)\}\{(letter)\}\{(id)\}\{(type)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the letter group title.

\bibglssetlettergrouptitlehier{\{(title)\}\{(letter)\}\{(id)\}\{(type)\}\{(parent)\}\{(level)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the hierarchical letter group title.

\bibglssetlocationrecordcount{\{entry-label\}\{(counter)\}\{(location)\}\{(value)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the location record count for the given entry.

\bibglssetmergedgrouptitle{\{(id)\}\{(type)\}\{(n)\}\{(g_1)\}\{(g_2)\}…\{(g_{n-1})\}\{(g_n)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the merged group title.

\bibglssetmergedgrouptitlehier{\{(id)\}\{(type)\}\{(n)\}\{(g_1)\}\{(g_2)\}…\{(g_{n-1})\}\{(g_n)\}\{(parent)\}\{(level)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the merged hierarchical group title.

\bibglssetnumbergrouptitle{\{(value)\}\{(id)\}\{(type)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the number group title.

\bibglssetnumbergrouptitlehier{\{(value)\}\{(id)\}\{(type)\}\{(parent)\}\{(level)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the hierarchical number group title.

\bibglssetothergrouptitle{\{(character)\}\{(id)\}\{(type)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the non-letter group title.

\bibglssetothergrouptitlehier{\{(character)\}\{(id)\}\{(type)\}\{(parent)\}\{(level)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the hierarchical non-letter group title.

\bibglssetrecordcount{\{entry-label\}\{(counter)\}\{(value)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the \texttt{(counter)} record count for the given entry.

\bibglssettimegrouptitle{\{(hh)\}\{(mm)\}\{(ss)\}\{(zone)\}\{(title)\}\{(group-id)\}\{(type)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the time (no date) group title.

\bibglssettimegrouptitlehier{\{(hh)\}\{(mm)\}\{(ss)\}\{(zone)\}\{(title)\}\{(group-id)\}\{(type)\}\{(parent)\}\{(level)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the hierarchical time (no date) group title.

\bibglssettotalrecordcount{\{entry-label\}\{(value)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the total record count for the given entry.

\bibglssetunicodegrouptitle{\{\{label\}\}\{(character)\}\{(id)\}\{(type)\}}  \hspace*{5.7cm} \texttt{bib2gls}

Sets the Unicode script, category or character code group title.

579
\bibglsetunicodelongtitlehier{{\{\langle label\rangle\}\{\langle character\rangle\}\{\langle id\rangle\}\{\langle type\rangle\}\{\langle parent\rangle\}\{\langle level\rangle\}}}bib2gls
Sets the Unicode script, category or character code hierarchical group title.

\bibglsetwidest{{\{\langle level\rangle\}\{\langle name\rangle\}}}bib2gls
Sets the widest name.

\bibglsetwidestfallback{{\{\langle glossary list\rangle\}}}bib2gls
Fallback used instead of \bibglsetwidest in the event that bib2gls can’t determine the widest name, where \langle glossary list\rangle is a comma-separated list of glossary labels.

\bibglsetwidestfortype{{\{\langle type\rangle\}\{\langle level\rangle\}\{\langle name\rangle\}}}bib2gls
Sets the widest name for the given glossary type.

\bibglsetwidestfortypefallback{{\{\langle type\rangle\}}}bib2gls
Fallback used instead of \bibglsetwidestfortype in the event that bib2gls can’t determine the widest name.

\bibglsetwidesttoplevelfallback{{\{\langle glossary list\rangle\}}}bib2gls
Fallback used instead of \bibglsetwidest in the event that bib2gls can’t determine the widest name where there are only top level entries, where \langle glossary list\rangle is a comma-separated list of glossary labels.

\bibglsetwidesttoplevelfortypefallback{{\{\langle type\rangle\}}}bib2gls
Fallback used instead of \bibglsetwidestfortype in the event that bib2gls can’t determine the widest name where there are only top-level entries.

\bibglssupplemental{\{\{n\}\{\langle list\rangle\}}}bib2gls
Supplemental list encapsulator.

\bibglssupplementalsep bib2gls
Separator between main and supplementary locations.

\bibglssupplementalsublist{\{\{n\}\{\langle external document\rangle\}\{\langle list\rangle\}}}bib2gls
Supplemental sub-list encapsulator.

\bibglssupplementalsubsep bib2gls
Separator between supplementary sub-lists.

\bibglstertiaryprefixlabel{\{\langle prefix\rangle\}}bib2gls
Hook provided to pick up the tertiary prefix, if required.

\bibglstime{\{\langle hour\rangle\}\{\langle minute\rangle\}\{\langle second\rangle\}\{\langle millisecond\rangle\}\{\langle dst\rangle\}\{\langle zone\rangle\}\{\langle original\rangle\}}}bib2gls
Used to markup a time converted from a field value.

\bibglstimegroup{\{\{hh\}\{\{mm\}\{\{ss\}\{\langle zone\rangle\}\{\langle title\rangle\}\{\langle group-id\rangle\}\{\langle type\rangle\}\}}}bib2gls
Expands to the time group label.

\bibglstimegrouphier{\{\{hh\}\{\{mm\}\{\{ss\}\{\langle zone\rangle\}\{\langle title\rangle\}\{\langle group-id\rangle\}\{\langle type\rangle\}\{\langle parent\rangle\}\{\langle level\rangle\}}}bib2gls
Expands to the hierarchical time group label.
\bibglstitlertime\{\langle hh \rangle\}\{\langle mm \rangle\}\{\langle ss \rangle\}\{\langle zone \rangle\}\{\langle title \rangle\}\{\langle group-id \rangle\}\{\langle type \rangle\}

Expands to the time group title.

\bibglstitlertimehier\{\langle hh \rangle\}\{\langle mm \rangle\}\{\langle ss \rangle\}\{\langle zone \rangle\}\{\langle title \rangle\}\{\langle group-id \rangle\}\{\langle type \rangle\}\{\langle parent \rangle\}\{\langle level \rangle\}

Expands to the hierarchical time group title.

\bibglstitlertitle\{\langle text \rangle\}

Converts \langle text \rangle to title case.

\bibglstitlecase\{\langle text \rangle\}

Expands to a literal underscore character.

\bibglstitlecase\{\langle text \rangle\}

Exports to the Unicode script or category or character code group label.

\bibglstitlecasel\{\langle label \rangle\}\{\langle character \rangle\}\{\langle id \rangle\}\{\langle type \rangle\}

Expands to the Unicode script or category or character code hierarchical group label.

\bibglstitlecasel\{\langle label \rangle\}\{\langle character \rangle\}\{\langle id \rangle\}\{\langle type \rangle\}\{\langle parent \rangle\}\{\langle level \rangle\}

Expands to the Unicode script or category or character code hierarchical group label.

\bibglssupercase\{\langle text \rangle\}

Converts \langle text \rangle to upper case.

\bibglssuppercase\{\langle text \rangle\}

Ensures that the given text is formatted according to the given category’s short format.

\bibglssusealias\{\langle label \rangle\}

Display the alias cross-reference for given entry.

\bibglssuseindex\{\langle label \rangle\}

Expands to the order of use index of the entry identified \langle label \rangle if \texttt{save-use-index} is set and the entry has records otherwise expands to empty.

\bibglssuselongfont\{\langle text \rangle\}\{\langle category \rangle\}

Ensures that the given text is formatted according to the given category’s long format.

\bibglssusesee\{\langle label \rangle\}

Display see cross-reference list for given entry.

\bibglssuseesalso\{\langle label \rangle\}

Display the seealso cross-reference list for given entry.

\bibliography\{\langle file list \rangle\}

Display bibliography created by \TeX.

\bigoperatornamefmt\{\langle text \rangle\}

Example command.
\boldsymbol{\langle symbol \rangle} \quad \text{amsmath}

Renders given maths symbol in bold if supported by the current font.

\bottomrule \quad \text{booktabs}

Horizontal rule for the bottom of a tabular-like environment.

C

\c{\langle character \rangle} \quad \text{kernel command}^*

Puts a cedilla accent over \langle character \rangle.

\capitalisewords{\langle text \rangle} \quad \text{mfistuc}^* \text{ v1.06+}

Converts the first letter of each word to upper case using \texttt{\makefirstuc}.

\caption{\langle list title \rangle}{\langle title \rangle} \quad \text{kernel command}

Caption title.

\CAT{\langle element-list \rangle} \quad \text{bib2gls quark}

A quark to denote a string concatenation (see section 5.1) in the conditional parts of \texttt{assign-fields}. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}.

\ce{\langle formula \rangle} \quad \text{mhchem}^*

Displays the chemical formula.

\chapter{\langle toc title \rangle}{\langle title \rangle} \quad \text{book or report classes}

Chapter heading.

\chapter*{\langle title \rangle} \quad \text{book or report classes}

Unnumbered chapter heading.

\char{\langle number \rangle} \quad \text{\LaTeX primitive}^*

Accesses the character identified by \langle number \rangle (use \texttt{\char{\langle hex \rangle}} if the number is hexadecimal).

\citation{\langle label \rangle} \quad \text{kernel command}

Written to the .aux file on each occurrence of \texttt{\cite}.

\cite{\langle label \rangle} \quad \text{kernel command}^*

Cross-reference a bibliographic citation.

\cjkname{\langle CJK characters \rangle} \quad \text{Example command.}

\color{\langle model \rangle}{\langle spec \rangle} \quad \text{color}

Switches the current font colour.

\CS{\langle element-list \rangle} \quad \text{bib2gls quark}

A quark to denote a control sequence element in \texttt{assign-fields}. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}. 
General Command Summary

\cs{⟨csname⟩}
Locally defined by \GlsXtrResourceInitEscSequences to expand to the literal string \csname when the resource options are written to the \texttt{.aux} file. This technically isn’t a \bib2gls quark, although it’s included in that category, as it’s not looked for by \bib2gls.

\csuse{⟨cs-name⟩}
etoolbox*
Uses the control sequence whose name is given by \langle cs-name \rangle or does nothing if the command isn’t defined.

\currentglossary
glossaries
Defined within the glossary to the current glossary type, this has no meaning outside of the glossary list.

\DeclareOptions{⟨name⟩}{⟨code⟩}
kernel command*
Declares an option with the given \langle name \rangle.

\DeclareOptions*{⟨code⟩}
kernel command*
Indicates what to do with unknown options.

\def{⟨cs⟩}{⟨syntax⟩}{⟨definition⟩}
\TeX primitive*
Defines the control sequence \langle cs \rangle, without checking if the command already exists.

\delimN
glossaries
Used to delimited individual locations.

\delimR
glossaries
Used as a separator between the start and end locations of a range.

\descriptionname
glossaries
Language-sensitive name used for the description header for some glossary styles.

\detokenize{⟨general text⟩}
\\ \ \ e-\TeX primitive*
Expands the argument to a list of character tokens.

\dGls[⟨options⟩]{⟨label⟩}{⟨insert⟩}
glossaries-extra-bib2gls v1.37+
Intended for documents with a mixture of single and dual entries, this is like \Gls but tries to determine the label prefix from the label prefix list.

\dgls[⟨options⟩]{⟨label⟩}{⟨insert⟩}
glossaries-extra-bib2gls v1.37+
Intended for documents with a mixture of single and dual entries, this is like \gls but tries to determine the label prefix from the label prefix list.

\dglsdisp[⟨options⟩]{⟨label⟩}{⟨text⟩}
glossaries-extra-bib2gls v1.37+
Like \glsdisp but tries the prefixes identified with commands like \glsxtraddlabelprefix.

\dglslink[⟨options⟩]{⟨label⟩}{⟨text⟩}
glossaries-extra-bib2gls v1.37+
Like \glslink but tries the prefixes identified with commands like \glsxtraddlabelprefix.
\Glspl{⟨options⟩}{⟨label⟩}{⟨insert⟩} \hspace{1cm} \textit{glossaries-extra-bib2gls v1.37+}

Intended for documents with a mixture of single and dual entries, this is like \Glspl but tries to determine the label prefix from the label prefix list.

\dglspl{⟨options⟩}{⟨label⟩}{⟨insert⟩} \hspace{1cm} \textit{glossaries-extra-bib2gls v1.37+}

Intended for documents with a mixture of single and dual entries, this is like \Glspl but tries to determine the label prefix from the label prefix list.

\DH \hspace{1cm} \textit{kernel command*}

Produces the upper case eth Ð.

\dh \hspace{1cm} \textit{kernel command*}

Produces the lower case eth ð.

\diamondsuit \hspace{1cm} \textit{kernel command* (maths mode)}

Diamond symbol (♦).

\displaystyle \hspace{1cm} \textit{kernel command (maths mode)}

Switch to display maths style.

\DJ \hspace{1cm} \textit{kernel command*}

Produces the upper case d-stroke Đ.

\dj \hspace{1cm} \textit{kernel command*}

Produces the lower case d-stroke đ.

\DTLandname \hspace{1cm} \textit{datatool-base v2.28+}

Used in the definition of \DTLlistformatlastsep.

\DTLformatlist{⟨list⟩} \hspace{1cm} \textit{datatool-base v2.28+}

Formats a comma-separated list.

\DTLlistformatlastsep \hspace{1cm} \textit{datatool-base v2.28+}

Used by \DTLformatlist to separate the last two items in the list.

\DTLlistformatoxford \hspace{1cm} \textit{datatool-base v2.28+}

Insert before \DTLlistformatlastsep if the list has three or more items.

\DTMdisplaydate{⟨year⟩}{⟨month⟩}{⟨day⟩}{⟨dow⟩} \hspace{1cm} \textit{datetime2}

Formats the given date where all arguments are numeric.

E

\edef{⟨cs⟩}{⟨syntax⟩}{⟨definition⟩} \hspace{1cm} \textit{TeX primitive*}

Defines the control sequence ⟨cs⟩ to the full expansion of ⟨definition⟩, without checking if the command already exists.

\eglsupdatewidest{⟨level⟩}{⟨text⟩} \hspace{1cm} \textit{glossaries-extra-stylemods v1.23+}

As \glsupdatewidest but expands ⟨text⟩.

\em \hspace{1cm} \textit{kernel command}

Switch to emphasized font (until end of current scope).
**General Command Summary**

\texttt{\textbackslash emph\{\text\}} \hfill kernel command

Emphasizes the given text (italic or slanted if the surrounding font is upright, otherwise upright font is used).

\texttt{\textbackslash endfoot} \hfill longtable

Ends the footer section.

\texttt{\textbackslash endhead} \hfill longtable

Ends the header section.

\texttt{\textbackslash ensuremath\{\text\}} \hfill kernel command*

Ensures the argument is in math mode. As a general rule this should only be used if you know for certain that the argument just contains mathematical markup and doesn’t cause a change in mode.

\texttt{\textbackslash FIRSTLC\{}\{element-list\}\}\hfill \text{bib2gls quark}

A quark to denote a first-letter lower case change in assign-fields syntax. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}.

\texttt{\textbackslash FIRSTUC\{}\{element-list\}\}\hfill \text{bib2gls quark}

A quark to denote a first-letter upper case change in assign-fields syntax. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}.

\texttt{\textbackslash footnote\{}\{number\}\{text\}\}\hfill kernel command

Displays the given text as a footnote.

\texttt{\textbackslash forall} \hfill kernel command* (maths mode)

For all symbol (∀).

\texttt{\textbackslash forallglossaries\{}\{glossary-list\}\{cs\}\{body\}\}\hfill \text{glossaries}

Iterates over all glossaries identified in the comma-separated \{glossary-list\} (or all defined non-ignored glossaries if the optional argument is omitted) and performs \{body\} where you can use the control sequence \{cs\} to reference the current glossary label.

\texttt{\textbackslash forallglsentries\{}\{glossary-list\}\{cs\}\{body\}\}\hfill \text{glossaries}

Iterates over all entries defined in the comma-separated \{glossary-list\} (or all defined non-ignored glossaries if the optional argument is omitted) and perform \{body\} where you can use the control sequence \{cs\} to reference the current entry label.

\texttt{\textbackslash forglsentries\{}\{type\}\{cs\}\{body\}\}\hfill \text{glossaries}

Iterates over all entries defined in the glossary identified by \{type\} (or the default, if \{type\} is omitted) and perform \{body\} where you can use the control sequence \{cs\} to reference the current entry label.

\texttt{\textbackslash forlistloop\{}\{handler cs\}\{list cs\}\}\hfill \text{etoolbox}

Iterates over the internal list given by the command \{list cs\} and performs \{handler \texttt{cs}\}\{element\} for each element.
\frontmatter
Switches to front matter.

G

\glolinkprefix
glossaries
Target name prefix used in entry hyperlinks.

\glossariesextrasetup\{\texttt{key=value list}\}
glossaries-extra
Applies the extension glossaries-extra options that are allowed to be changed after the package has loaded.

\glossaryheader
glossaries
Implemented at the start of a glossary (modified by glossary styles).

\glossaryname
glossaries or language packages
Language-sensitive name used for the title of the default main glossary.

\glossarypostamble
glossaries
The postamble that’s placed after each glossary.

\glossarypreamble
glossaries
The preamble for all glossaries except those that have the preamble explicitly set with \apptoglossarypreamble.

\glossentry\{\texttt{label}\}\{\texttt{location list}\}
glossaries v3.08a+
Used in the glossary to display a top-level entry.

\Glossentrydesc\{\texttt{label}\}
glossaries
Like \glossentrydesc but converts the first letter to upper case.

\glossentrydesc\{\texttt{label}\}
glossaries
Used by glossary styles to display the description.

\Glossentryname\{\texttt{label}\}
glossaries
Like \glossentryname but converts the first letter to upper case.

\glossentryname\{\texttt{label}\}
glossaries
Used by glossary styles to display the name.

\glossentrynameother\{\texttt{label}\}\{\texttt{field}\}
glossaries-extra v1.22+
Acts like \glossentryname (obeys glossname and glossnamefont or \glsnamefont and the post-name hook) but uses the given \texttt{field} instead of the \texttt{name} field.

\Glossentrysymbol\{\texttt{label}\}
glossaries
Like \glossentrysymbol but converts the first letter to upper case.

\glossentrysymbol\{\texttt{label}\}
glossaries
Used by glossary styles to display the symbol.

\glossxtrsetpopts
glossaries-extra v1.07+
Glossary hook that uses \glsxtrsetpopts to enable hyperlinks by default for \glsxtrp.
\gls{(options)}{(label)}{(insert)}
glossaries
As \texttt{gls} but converts the link text to upper case.
\Gls{(options)}{(label)}{(insert)}
glossaries
As \texttt{gls} but converts the first letter of the link text to upper case.
\gl{(options)}{(label)}{(insert)}
glossaries
On first use displays the first use text (the value of the \texttt{first} field for general entries) and on subsequent use displays the subsequent use text (the value of the \texttt{text} field for general entries) where the text is optionally hyperlinked to the relevant place in the glossary. The options are the same as for \texttt{glslink}.
\glsabbrvdefaultfont{(text)}
glossaries-extra
Used by the abbreviation styles that don’t have a specific font to format the short form. The default definition just does its argument without any formatting.
\glsabbrvemfont{(text)}
glossaries-extra v1.04+
Used with “em” abbreviation styles to format the short form using \texttt{\textit}.\glsabbrvfont{(text)}
glossaries-extra
Generic abbreviation font command for the short form.
\glsabbrvhyphenfont{(text)}
glossaries-extra v1.17+
Used by the “hyphen” abbreviation styles to format the short form.
\glsabbrvonlyfont{(text)}
glossaries-extra v1.17+
Used with “only” abbreviation styles to format the short form. The default definition just uses \texttt{\glsabbrvdefaultfont}.
\glsabbrvscfont{(text)}
glossaries-extra v1.17+
Used with “sc” abbreviation styles to format the short form using \texttt{\textsc}.
\glsabbrvsmfont{(text)}
glossaries-extra v1.17+
Used with “sm” abbreviation styles to format the short form using \texttt{\textsmaller}.
\glsabbrvuserfont{(text)}
glossaries-extra v1.04+
Used with “user” abbreviation styles to format the short form. The default definition just uses \texttt{\glsabbrvdefaultfont}.
\Glsaccessdesc{(label)}
glossaries-extra*
Expands to the value of the \texttt{description} field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \texttt{\glsentrydesc}).
\glsaccessdesc{(label)}
glossaries-extra*
Expands to the value of the \texttt{description} field with the accessibility support for that field, if provided (otherwise behaves the same as \texttt{\glsentrydesc}).
\Glsaccessdescplural{(label)}
glossaries-extra*
Expands to the value of the \texttt{descriptionplural} field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \texttt{\glsentrydescplural}).
\glsaccessdescplural{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the descriptionplural field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentrydescplural).

\glsaccessdisplay{⟨field⟩}{⟨text⟩}{⟨label⟩}\nop{\texttt{glossaries-access}}

Displays ⟨text⟩ with the accessibility support provided by \glsentry{field}{access}{⟨label⟩}.

\Glsaccessfirst{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the first field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentryfirst).

\glsaccessfirst{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the first field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryfirst).

\Glsaccessfirstplural{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the firstplural field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentryfirstplural).

\glsaccessfirstplural{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the firstplural field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryfirstplural).

\Glsaccesslong{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the long field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentrylong).

\glsaccesslong{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the long field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentrylong).

\Glsaccesslongpl{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the longplural field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentrylongpl).

\glsaccesslongpl{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the longplural field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentrylongpl).

\Glsaccessname{⟨label⟩}\nop{\texttt{glossaries-extra}}

Expands to the value of the name field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentryname).
\glsaccessname{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the name field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryname).

\Glsaccessplural{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the plural field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentryplural).

\glsaccessplural{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the plural field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryplural).

\Glsaccessshort{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the short field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentryshort).

\glsaccessshort{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the short field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryshort).

\Glsaccessshortpl{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the shortplural field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentryshortpl).

\glsaccessshortpl{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the shortplural field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryshortpl).

\Glsaccesssymbol{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the symbol field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentrysymbol).

\glsaccesssymbol{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the symbol field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentrysymbol).

\Glsaccesssymbolplural{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the symbolplural field with the first letter converted to upper case and with the accessibility support for that field, if provided (otherwise behaves the same as \Glsentrysymbolplural).

\glsaccesssymbolplural{⟨label⟩} \textit{glossaries-extra*}
Expands to the value of the symbolplural field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentrysymbolplural).
General Command Summary

`\Glsaccesstext\{\langle label\rangle\}` \textit{glossaries-extra}^*
Expands to the value of the \texttt{text} field with the first letter converted to upper case and
with the accessibility support for that field, if provided (otherwise behaves the same as `\glsentrytext`).

`\glsaccesstext\{\langle label\rangle\}` \textit{glossaries-extra}^*
Expands to the value of the \texttt{text} field with the accessibility support for that field, if
provided (otherwise behaves the same as `\glsentrytext`).

`\glsaccsupp\{\langle accessible text\rangle\}\{\langle text\rangle\}` \textit{glossaries-accsupp}
Used by the accessibility support to interface with the accsupp package (use \texttt{\xglsaccsupp} if \texttt{\langle text\rangle} needs to be fully expanded first).

`\glsadd\{\langle options\rangle\}\{\langle label\rangle\}` \textit{glossaries}
Indexes the entry without displaying any text.

\textbf{Options:}

- \texttt{counter=\{\langle counter-name\rangle\}} \textit{glossaries}
  Sets the counter to use for the record

- \texttt{format=\{\langle encap\rangle\}} \textit{glossaries}
  Sets the \texttt{ENCAP} for the record to \texttt{\langle encap\rangle}, optionally with the start or end range markers

- \texttt{theVvalue=\{\langle value\rangle\}} \textit{glossaries-extra v1.14+}
  The hyperlink target corresponding to the value of \texttt{\langle value\rangle}, if appropriate

- \texttt{theVvalue=\{\langle value\rangle\}} \textit{glossaries-extra v1.14+}
  Overrides the record value so that it’s the given \texttt{\langle value\rangle} not obtained from the
  associated counter

`\glsaddall\{\langle options\rangle\}` \textit{glossaries}
Iterates over all entries defined for all glossaries (or for the sub-list provided by
\texttt{\langle options\rangle}) and performs `\glsadd\{\langle options\rangle\}` for each entry. This
command isn’t suitable for use with \texttt{\bib2gls}. Use the \textit{selection} option instead.

`\glsaddallunused\{\langle list\rangle\}` \textit{glossaries}
Iterates over all entries defined for all glossaries (or for the sub-list provided in the
options) and performs `\glsadd` for each entry that hasn’t been used with the \texttt{\langle options\rangle}
set to \texttt{\glsignore}. This command isn’t suitable for use with \texttt{\bib2gls}. Use the \textit{selection}
option instead.

`\glsaddeach\{\langle options\rangle\}\{\langle label list\rangle\}` \textit{glossaries-extra v1.31+}
Indexes each entry identified in the comma-separated list of labels without displaying
any text.

`\glsaddkey\{\langle key\rangle\}\{\langle default value\rangle\}\{\langle no link cs\rangle\}\{\langle link allcaps cs\rangle\}` \textit{glossaries}
Adds a new key for use in `\newglossaryentry` and associated commands to access it.

`\glsaddstoragekey\{\langle key\rangle\}\{\langle default value\rangle\}\{\langle no link cs\rangle\}` \textit{glossaries}
Adds a new key for internal use that can be set in `\newglossaryentry`.
\glsautoprefix\textit{glossaries}\n
Prefix used for the automatically labelling triggered by the \numberedsection={autolabel} option.

\glsbackslash\textit{glossaries}\n
Expands to a literal backslash \ character.

\glscapturedgroup\textit{glossaries-extra-bib2gls v1.31+}\n
Expands to \string$\textit{g}$.

\glscategory\langle label\rangle\textit{glossaries-extra}\n
Expands to the value of the \textit{category} field for the entry identified by \langle label\rangle or nothing if the entry hasn’t been defined.

\glsclosebrace\textit{glossaries}\n
Expands to a literal close brace } character.

\glscurrententrylabel\textit{glossaries}\n
Only for use in the glossary, such as in the style or in the post-name or post-description hooks, this expands to the label of the current entry.

\glscurrentfieldvalue\textit{glossaries}\n
Only for use in the \langle true\rangle part of \textit{ifgls}hashfield or \textit{glsxtrifhashfield}, this expands to the field value.

\glsdefaulttype\textit{glossaries}\n
The default glossary type.

\glsdefpostdesc\langle category\rangle\langle definition\rangle\textit{glossaries-extra v1.31+}\n
Define the post-description hook \textit{glsxtrpostdesc}\langle category\rangle for the given category.

\glsdefpostlink\langle category\rangle\langle definition\rangle\textit{glossaries-extra v1.31+}\n
Define the post-link hook \textit{glsxtrpostlink}\langle category\rangle for the given category.

\glsdefpostname\langle category\rangle\langle definition\rangle\textit{glossaries-extra v1.31+}\n
Define the post-name hook \textit{glsxtrpostname}\langle category\rangle for the given category.

\glsdesc\langle options\rangle\langle label\rangle\langle insert\rangle\textit{glossaries}\n
Links to the entry’s definition in the glossary with the link text obtained from the \textit{description} field without altering the first use flag.

\glsdescriptionaccessdisplay\langle text\rangle\langle label\rangle\textit{glossaries-access}\n
Displays \langle text\rangle with the accessibility support provided by \textit{gls}entrydescaccess\langle label\rangle.

\glsdescriptionpluralaccessdisplay\langle text\rangle\langle label\rangle\textit{glossaries-access}\n
Displays \langle text\rangle with the accessibility support provided by \textit{gls}entrydescpluralaccess\langle label\rangle.

\glsdescwidth\textit{glossary-long and glossary-super}\n
Length register used by the tabular styles to specify the width of the description column.
Disables the creation of hyperlinks and targets for the glossary commands that support them (automatically implemented if \hyperref isn’t loaded before glossaries).

Links to the entry’s definition in the glossary with the given link text and marks the entry as having been used. The options are the same as for \glslink.

If the entry given by \langle label \rangle exists, \langle code \rangle is done, otherwise an error (or warning with glossaries-extra’s undefaction={warn} option) is triggered.

If the entry given by \langle label \rangle doesn’t exist, \langle code \rangle is done, otherwise an error (or warning with glossaries-extra’s undefaction={warn} option) is triggered and \langle else code \rangle is done.

Enables the creation of hyperlinks and targets for the glossary commands that support them (automatically implemented if \hyperref is loaded before glossaries).

Essentially like \glsaddeach[format={}), \langle options \rangle \} \{ \langle label-list \rangle \}. If format is used in \langle options \rangle, the close marker ) will be inserted in front of the value.

Expands to the value of the access field.

Governs the way the glossaryentry counter is displayed by \glsentryitem.

Used as a prefix in the \label command automatically implemented by the entrycounter and subentrycounter options.

Displays the value of the description field with the first letter converted to upper case.

Expands to the value of the description field.

Expands to the value of the descriptionaccess field.
\Glsentrydescplural{⟨label⟩} \hfill glossaries*
Displays the value of the descriptionplural field with the first letter converted to upper case.

\glsentrydescplural{⟨label⟩} \hfill glossaries*
Expands to the value of the descriptionplural field.

\glsentrydescpluralaccess{⟨label⟩} \hfill glossaries-access
Expands to the value of the descriptionpluralaccess field.

\Glsentryfirst{⟨label⟩} \hfill glossaries*
Displays the value of the first field with the first letter converted to upper case.

\glsentryfirst{⟨label⟩} \hfill glossaries*
Expands to the value of the first field.

\glsentryfirstaccess{⟨label⟩} \hfill glossaries-access
Expands to the value of the firstaccess field.

\Glsentryfirstplural{⟨label⟩} \hfill glossaries*
Displays the value of the firstplural field with the first letter converted to upper case.

\glsentryfirstplural{⟨label⟩} \hfill glossaries*
Expands to the value of the firstplural field.

\glsentryfirstpluralaccess{⟨label⟩} \hfill glossaries-access
Expands to the value of the firstpluralaccess field.

\Glsentryitem{⟨label⟩} \hfill glossaries v3.0+
Increments and displays the glossaryentry counter, if appropriate.

\Glsentrylong{⟨label⟩} \hfill glossaries*
Displays the value of the long field without any formatting or indexing but with the first letter converted to upper case.

\glsentrylong{⟨label⟩} \hfill glossaries*
Expands to the value of the long field without any formatting or indexing.

\glsentrylongaccess{⟨label⟩} \hfill glossaries-access
Expands to the value of the longaccess field.

\Glsentrylongpl{⟨label⟩} \hfill glossaries*
Displays the value of the longplural field without any formatting or indexing but with the first letter converted to upper case.

\glsentrylongpl{⟨label⟩} \hfill glossaries*
Expands to the value of the longplural field without any formatting or indexing.

\glsentrylongpluralaccess{⟨label⟩} \hfill glossaries-access
Expands to the value of the longpluralaccess field.

\Glsentryname{⟨label⟩} \hfill glossaries*
Displays the value of the name field with the first character converted to upper case.
\glsentryname{⟨label⟩} \hspace{1cm} glossaries*
Expands to the value of the name field.

\Glsentryplural{⟨label⟩} \hspace{1cm} glossaries*
Displays the value of the plural field with the first letter converted to upper case.

\glsentryplural{⟨label⟩} \hspace{1cm} glossaries*
Expands to the value of the plural field.

\glsentrypluralaccess{⟨label⟩} \hspace{1cm} glossaries-access
Expands to the value of the pluralaccess field.

\Glsentryprefix{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefix field with the first letter converted to upper case.

\glsentryprefix{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefix field.

\Glsentryprefixfirst{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefixfirst field with the first letter converted to upper case.

\glsentryprefixfirst{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefixfirst field.

\Glsentryprefixfirstplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefixfirstplural field with the first letter converted to upper case.

\glsentryprefixfirstplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefixfirstplural field.

\Glsentryprefixplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefixplural field with the first letter converted to upper case.

\glsentryprefixplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the prefixplural field.

\Glsentryshort{⟨label⟩} \hspace{1cm} glossaries*
Displays the value of the short field without any formatting or indexing but with the first letter converted to upper case.

\glsentryshort{⟨label⟩} \hspace{1cm} glossaries*
Expands to the value of the short field without any formatting or indexing.

\glsentryshortaccess{⟨label⟩} \hspace{1cm} glossaries-access
Expands to the value of the shortaccess field.

\Glsentryshortpl{⟨label⟩} \hspace{1cm} glossaries*
Displays the value of the shortplural field without any formatting or indexing but with the first letter converted to upper case.

594
\glsentryshortpl\{\textit{label}\}\rulenotapplicable{glossaries}*
Expands to the value of the \texttt{shortplural} field without any formatting or indexing.
\glsentryshortpluralaccess\{\textit{label}\}\rulenotapplicable{glossaries-access}
Expands to the value of the \texttt{shortpluralaccess} field.
\Glsentrysymbol\{\textit{label}\}\rulenotapplicable{glossaries}*
Displays the value of the \texttt{symbol} field with the first letter converted to upper case.
\glsentrysymbol\{\textit{label}\}\rulenotapplicable{glossaries}*
Expands to the value of the \texttt{symbol} field.
\glsentrysymbolaccess\{\textit{label}\}\rulenotapplicable{glossaries-access}
Expands to the value of the \texttt{symbolaccess} field.
\Glsentrysymbolplural\{\textit{label}\}\rulenotapplicable{glossaries}*
Displays the value of the \texttt{symbolplural} field with the first letter converted to upper case.
\glsentrysymbolplural\{\textit{label}\}\rulenotapplicable{glossaries}*
Expands to the value of the \texttt{symbolplural} field.
\glsentrysymbolpluralaccess\{\textit{label}\}\rulenotapplicable{glossaries-access}
Expands to the value of the \texttt{symbolpluralaccess} field.
\Glsentrytext\{\textit{label}\}\rulenotapplicable{glossaries}*
Displays the value of the \texttt{text} field with the first letter converted to upper case.
\glsentrytext\{\textit{label}\}\rulenotapplicable{glossaries}*
Expands to the value of the \texttt{text} field.
\glsentrytextaccess\{\textit{label}\}\rulenotapplicable{glossaries-access}
Expands to the value of the \texttt{textaccess} field.
\glsentrytitlecase\{\textit{entry label}\}\{\textit{field label}\}\rulenotapplicable{glossaries} v4.22+
Fetches the given field and applies \texttt{\textcap} to it.
\Glsentryuseri\{\textit{label}\}\rulenotapplicable{glossaries}*
Displays the value of the \texttt{user1} field with the first letter converted to upper case.
\glsentryuseri\{\textit{label}\}\rulenotapplicable{glossaries}*
Expands to the value of the \texttt{user1} field.
\Glsentryuserii\{\textit{label}\}\rulenotapplicable{glossaries}*
Displays the value of the \texttt{user2} field with the first letter converted to upper case.
\glsentryuserii\{\textit{label}\}\rulenotapplicable{glossaries}*
Expands to the value of the \texttt{user2} field.
\Glsentryuseriii\{\textit{label}\}\rulenotapplicable{glossaries}*
Displays the value of the \texttt{user3} field with the first letter converted to upper case.
\glsentryuseriii\{\textit{label}\}\rulenotapplicable{glossaries}*
Expands to the value of the \texttt{user3} field.
General Command Summary

\Glsentryuseriv{⟨label⟩} \hspace{1cm} \text{glossaries*}
Displays the value of the user4 field with the first letter converted to upper case.

\glsentryuseriv{⟨label⟩} \hspace{1cm} \text{glossaries*}
Expands to the value of the user4 field.

\Glsentryuseriv{⟨label⟩} \hspace{1cm} \text{glossaries*}
Displays the value of the user5 field with the first letter converted to upper case.

\glsentryuseriv{⟨label⟩} \hspace{1cm} \text{glossaries*}
Expands to the value of the user5 field.

\Glsentryuserv{⟨label⟩} \hspace{1cm} \text{glossaries*}
Displays the value of the user6 field with the first letter converted to upper case.

\glsentryuserv{⟨label⟩} \hspace{1cm} \text{glossaries*}
Expands to the value of the user6 field.

\glsexpandfields \hspace{1cm} \text{glossaries}
Switches on field expansion.

\glsextrapostnamehook{⟨label⟩} \hspace{1cm} \text{glossaries-extra v1.25+}
Additional category-independent code for the post-name hook.

\glsfielddef{⟨entry label⟩}{⟨field label⟩}{⟨definition⟩} \hspace{1cm} \text{glossaries v4.16+}
Changes the value of the given entry’s field to ⟨definition⟩ (localised by any scope).

\glsfieldedef{⟨entry label⟩}{⟨field label⟩}{⟨definition⟩} \hspace{1cm} \text{glossaries v4.16+}
Changes the value of the given entry’s field to the full expansion of ⟨definition⟩ (localised by any scope).

\glsfieldfetch{⟨label⟩}{⟨field⟩}{⟨cs⟩} \hspace{1cm} \text{glossaries v4.16+}
Fetches the value of the given field for the given label and stores it in the command ⟨cs⟩.

\glsfieldgdef{⟨entry label⟩}{⟨field label⟩}{⟨definition⟩} \hspace{1cm} \text{glossaries v4.16+}
Globally changes the value of the given entry’s field to ⟨definition⟩.

\glsfieldxdef{⟨entry label⟩}{⟨field label⟩}{⟨definition⟩} \hspace{1cm} \text{glossaries v4.16+}
Globally changes the value of the given entry’s field to the full expansion of ⟨definition⟩.

\glsFindWidestLevelTwo{⟨glossary list⟩} \hspace{1cm} \text{glossaries-extra-stylemods}
Finds the widest name in the given glossaries for the top level and first two sub-levels.

\glsFindWidestTopLevelName{⟨glossary list⟩} \hspace{1cm} \text{glossaries-extra-stylemods}
CamelCase synonym for \glsfindwidesttoplevelname.

\glsfindwidesttoplevelname{⟨glossary list⟩} \hspace{1cm} \text{glossary-tree}
Finds the widest top-level name in the given glossaries.

\Glsfirst{⟨options⟩}{⟨label⟩}{⟨insert⟩} \hspace{1cm} \text{glossaries}
As \glsfirst but converts the first letter to upper case.

\glsfirst{⟨options⟩}{⟨label⟩}{⟨insert⟩} \hspace{1cm} \text{glossaries}
Links to the entry’s definition in the glossary with the link text obtained from the first field without altering the first use flag.
General Command Summary

\glsfirstabbrvdefaultfont\{\langle text \rangle\}  
Used by the abbreviation styles that don’t have a specific font to format the short form on first use. The default definition uses \glsabbrvdefaultfont.

\glsfirstabbrvemfont\{\langle text \rangle\}  
Used with “em” abbreviation styles to format the short form on first use. This defaults to \glsabbrvemfont.

\glsfirstabbrvhypenfont\{\langle text \rangle\}  
Used by the “hyphen” abbreviation styles to format the short form on first use.

\glsfirstabbrvonlyfont\{\langle text \rangle\}  
Used with “only” abbreviation styles to format the short form on first use. The default definition just uses \glsabbrvonlyfont.

\glsfirstabbrvscfont\{\langle text \rangle\}  
Used with “sc” abbreviation styles to format the short form on first use. This defaults to \glsabbrvscfont.

\glsfirstabbrvsmfont\{\langle text \rangle\}  
Used with “sm” abbreviation styles to format the short form on first use. This defaults to \glsabbrvsmfont.

\glsfirstabbrvusertemplate\{\langle text \rangle\}  
Used with “user” abbreviation styles to format the short form on first use. The default definition just uses \glsabbrvusertemplate.

\glsfirstaccessdisplay\{\langle text \rangle\}\{\langle label \rangle\}  
Displays \langle text \rangle with the accessibility support provided by \glsentryfirstaccess\{\langle label \rangle\}.

\glsfirstlongdefaultfont\{\langle text \rangle\}  
Used by the abbreviation styles that don’t have a specific font to format the long form on first use. The default definition uses \glslongdefaultfont.

\glsfirstlongemfont\{\langle text \rangle\}  
Used with “long-em” abbreviation styles to format the long form on first use. This defaults to \glslongemfont.

\glsfirstlongfootnotefont\{\langle text \rangle\}  
Used with the “footnote” abbreviation styles to format the long form on first use.

\glsfirstlonghyphenfont\{\langle text \rangle\}  
Used by the “hyphen” abbreviation styles to format the long form on first use.

\glsfirstlongonlyfont\{\langle text \rangle\}  
Used with “only” abbreviation styles to format the long form on first use. The default definition just uses \glslongonlyfont.

\glsfirstlonguserfont\{\langle text \rangle\}  
Used with “user” abbreviation styles to format the long form on first use. The default definition just uses \glslonguserfont.
General Command Summary

\glsfirstpluralaccessdisplay{{text}}{\{label\}}
glossaries-access
Displays \textit{\textlangle text\textrangle} with the accessibility support provided by
\glsentryfirstpluralaccess{\{label\}}.

\glsfmtfirst{\{label\}}
glossaries-extra
Provided for use in section or caption titles, this displays the given entry’s \textit{first} field.

\glsfmtfull{\{label\}}
glossaries-extra
Provided for use in section or caption titles, this displays the full form of the given abbreviation (using the inline style that matches \glsxtrfull).

\glsfmtlong{\{label\}}
glossaries-extra
Provided for use in section or caption titles, this displays the long form of the given abbreviation.

\glsfmtname{\{label\}}
glossaries-extra
Provided for use in section or caption titles, this displays the given entry’s name.

\glsfmtshort{\{label\}}
glossaries-extra
Provided for use in section or caption titles, this displays the short form of the given abbreviation.

\glsfmttext{\{label\}}
glossaries-extra
Provided for use in section or caption titles, this displays the given entry’s \textit{text} field.

\glsgroupheading{\{label\}}
glossaries
Formats the heading for the group identified by the given label.

\glsgroupskip
glossaries
Inserted between groups to create some vertical spacing (this command is modified by
glossary styles, and may be switched off with the \textit{nogroupskip} option).

\glsshashchar
glossaries-extra-bib2gls* v1.49+
Expands to a literal hash character \texttt{#}.

\glshex
glossaries-extra v1.21+ (moved to glossaries-extra-bib2gls in v1.27)
Expands to \texttt{\string\u}.

\glshyperlink{{\{link text\}}}{\{label\}}
glossaries*
Creates a hyperlink to the entry information in the glossary.

\glshypernumber{\{text\}}
glossaries
A location format that has a hyperlink (if enabled).

\glsifcategory{\{label\}}{\{category\}}{\{true\}}{\{false\}}
glossaries-extra
Does \textit{\textlangle true\textrangle} if the \textit{category} field for the entry given by \textit{\textlangle label\textrangle} is \textit{\textlangle category\textrangle}.

\glsignore{\{text\}}
glossaries
Does nothing but when used as a location format \texttt{bib2gls} recognises it as an ignored record.

\glsinlinedescformat{{\{description\}}}{\{symbol\}}{\{location list\}}
glossary-inline v3.03+
Format’s the entry’s description, symbol and location list. This ignores the symbol and
location by default.
General Command Summary

\glsinlinedopostchild

Group headings aren’t supported by default, but if they are required, this command should be added to start of the definition of \glsgroupheading in case a heading follows a child entry.

\glsinlinenameformat\{\langle\text{label}\rangle\}\{\langle\text{name}\rangle\}

Format’s the entry’s name including target, if supported.

\glsinlineparentchildseparator

Separator between parent and child entries.

\glsinlinepostchild

Hook between child and next entry.

\glsinlineseparator

Separator between entries.

\glsinlinesubseparator

Separator between sub-entries.

\glslabel

Only for use in the post-link hooks, this expands to the label of the entry that was last referenced.

\glslink\{\langle\text{options}\rangle\}\{\langle\text{label}\rangle\}\{\langle\text{text}\rangle\}

Links to the entry’s definition in the glossary with the given link text without altering the first use flag.

Options:

counter=\{\langle\text{counter-name}\rangle\}

Sets the counter to use for the record

format=\{\langle\text{encap}\rangle\}

Sets the ENCAP for the record to \langle\text{encap}\rangle, optionally with the start or end range markers

hyper=\{\langle\text{boolean}\rangle\}

Indicates whether or not to make a hyperlink to the relevant glossary entry

hyperoutside=\{\langle\text{boolean}\rangle\}

Determines whether hyperlink should be outside of \glstextformat (default hyperoutside=true) or inside (hyperoutside=false)

local=\{\langle\text{boolean}\rangle\}

If true indicates to use \glslocalunset instead of the default global \glsunset to unset the first use flag

noindex=\{\langle\text{boolean}\rangle\}

Indicates whether or not to suppress indexing

prefix=\{\langle\text{label}\rangle\}

Locally changes \glolinkprefix to the given \langle\text{label}\rangle

textformat=\{\langle\text{cs-name}\rangle\}

If set, replaces \glstextformat with the command given by the control sequence name \langle\text{cs-name}\rangle to format the link text
The hyperlink target corresponding to the value of `\texttt{thevalue}`, if appropriate

Overrides the record value so that it’s the given \texttt{(value)} not obtained from the associated counter

Indicates whether to write the glossary information before (\texttt{wrgloss=before}) or after (\texttt{wrgloss=after}) the link text (default: \texttt{before})

Locally resets the first use flag so that the entry is marked as not used.

Locally unsets the first use flag so that the entry is marked as having been used.

Displays \texttt{(text)} with the accessibility support provided by \texttt{\glsentrylongaccess{\texttt{(label)}}}.

Used by the abbreviation styles that don’t have a specific font to format the long form.

The default definition just does its argument without any formatting.

Used with “long-em” abbreviation styles to format the long form using \texttt{\emph{}}.

Used with the styles provided by the glossary-longextra package to set the widest entry name.

As \texttt{\glslongextraSetWidest{\texttt{(text)}}} but only sets if \texttt{(text)} is wider than the current value.

Generic abbreviation font command for the long form.

Used with the “footnote” abbreviation styles to format the long form.

Used by the “hyphen” abbreviation styles to format the long form.

Used with “only” abbreviation styles to format the long form. The default definition just uses \texttt{\glslongdefaultfont{\texttt{(text)}}}.

Displays \texttt{(text)} with the accessibility support provided by \texttt{\glsentrylongpluralaccess{\texttt{(label)}}}.
\glslongtok\ glossaries
Token register used in the construction of acronyms or abbreviations to allow the style hooks to access the long form.

\glslonguserfont\{\text\}\ glossaries-extra v1.04+
Used with “user” abbreviation styles to format the long form. The default definition just uses \glslongdefaultfont.

\glslowercase\{\text\}\ glossaries v4.50+*
Converts \text to lower case.

\glnamespace[\{options\}][\{label\}][\{insert\}]\ glossaries
Links to the entry’s definition in the glossary with the link text obtained from the name field without altering the first use flag.

\glsnameaccessdisplay\{\text\}[\{label\}]\ glossaries-access
Displays \text with the accessibility support provided by \glsentryaccess{\{label\}}.

\glslongdefaultfont\{\text\}\ glossaries
Used by \glossentryname to format the name.

\glslongmessage\{\text\}\ glossaries
Language-sensitive name used for the numbers group and also used for the title of the glossary created with the numbers package option.

\glsoopenbrace\ glossaries*
Expands to a literal open brace { character.

\glsnamefont\{\text\}\ glossaries
Used by \glossentrynamefont to format the name.

\glsnavhypertarget[\{type\}][\{label\}][\{text\}]\ glossary-hypernav
Creates a hyper target for the group given by \{label\} for the given glossary type and uses \{text\} for the hyperlink text.

\glsnoexpandfields\ glossaries
Switches off field expansion.

\glsnoidxdisplayloc[\{prefix\}][\{counter\}][\{format\}][\{location\}]\ glossaries v4.04+
Used to display a regular location in the location field (with a hyperlink, if enabled).

\glsnoidxloclist\{\location list cs\}\ glossaries
Iterates over the given internal location list using the \glsnoidxloclisthandler handler.

\glsnoidxloclisthandler\{\location\}\ glossaries
The handler used by the internal list loop function used in \glsnoidxloclist.

\glsnumberformat\{\text\}\ glossaries
Default location format, uses \glshypernumber if hyperlinks enabled otherwise just does \text.

\glsnumbertabularx\ glossaries
Preamble command that will patch the tabularx environment to deal with the problem of unsetting the first use flag either explicitly with \glsunset or implicitly through commands like \gls (does nothing if tabularx hasn’t been loaded).
General Command Summary

\glpercentchar \textit{glossaries}*
Expands to a literal percent character $\%$ character.

\GLSpl[⟨options⟩]{⟨label⟩}{⟨insert⟩} \textit{glossaries}
As \texttt{GLS} but shows the plural form.

\Glsp[⟨options⟩]{⟨label⟩}{⟨insert⟩} \textit{glossaries}
As \texttt{gls} but shows the plural form.

\glsp[⟨options⟩]{⟨label⟩}{⟨insert⟩} \textit{glossaries}
As \texttt{gls} but shows the plural form.

\glsplural[⟨options⟩]{⟨label⟩}{⟨insert⟩} \textit{glossaries}
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{plural} field without altering the first use flag.

\glspluralaccessdisplay{⟨text⟩}{⟨label⟩} \textit{glossaries-access}
Displays ⟨text⟩ with the accessibility support provided by \glssentrypluralaccess{⟨label⟩}.

\glpluralaccessdisplay \textit{glossaries-access}
The suffix used to construct the default plural.

\glspost-inline \textit{glossary-inline v3.03+}
Glossary terminator.

\glspostdescription \textit{glossaries and modified by glossaries-extra}
A hook added after the description in some glossary styles (all if the \texttt{glossaries-extra-stylemods} package is loaded to patch them). This hook is used to reflect the \texttt{nopostdot} package option for \texttt{glossaries} and the \texttt{postpunc} option for \texttt{glossaries-extra}.

\glsp{⟨label⟩} \textit{glossaries-extra v1.07+}
Shortcut for \texttt{\glsxtrp{short}{⟨label⟩}}.

\glst{⟨label⟩} \textit{glossaries-extra v1.07+}
Shortcut for \texttt{\glsxtrp{text}{⟨label⟩}}.

\glquote{⟨text⟩} \textit{glossaries}
Encapsulates ⟨text⟩ with literal straight double-quotes "⟨text⟩".

\glssrefentry{⟨label⟩} \textit{glossaries v3.0+}
When used with \texttt{entrycounter} or \texttt{subentrycounter} may be used to cross-reference the entry’s number in the glossary list with \texttt{\ref}.

\glsrnewcommand{⟨cs⟩}{⟨n⟩}{⟨def⟩}{⟨code⟩} \textit{glossaries-extra-bib2gls v1.37+}
Behaves like \texttt{\renewcommand} but only generates a warning rather than an error if the command isn’t already defined.

\glssreset{⟨label⟩} \textit{glossaries}
Resets the first use flag so that the entry is marked as not used.

\glssresetentrycounter \textit{glossaries}
Resets the glossary entry counter if the \texttt{entrycounter} setting is on.
General Command Summary

\glssee{\langle \text{tag} \rangle}{\langle \text{label} \rangle}{\langle \text{xr label list} \rangle}
glossaries
Indexes a “see” cross-reference.

\glsseefirstitem{\langle \text{label} \rangle}
glossaries-extra v1.47+
As \glsseeitem but is used for the first label in the list. This just does \glsseeitem by default.

\glsseeformat{\langle \text{tag} \rangle}{\langle \text{labels} \rangle}{\langle \text{location (ignored)} \rangle}
glossaries
Formats the entries identified in the comma separated list of labels as a set of cross-references. This just does the tag (emphasized) followed by \glsseelist{\langle \text{labels} \rangle}.

\glsseeitem{\langle \text{label} \rangle}
glossaries
Formats an element of the cross-reference list. The default behaviour is to create a hyperlink (if enabled) to the referenced entry with the link text given by \glsseeitemformat{\langle \text{label} \rangle}.

\glsseeitemformat{\langle \text{label} \rangle}
glossaries v3.0+
Formats an element of the cross-reference list. With the base glossaries package this just does \glsentrytext{\langle \text{label} \rangle}. With glossaries-extra this uses either \glsentryshort or \glsentryname depending on whether or not the short field has been set.

\glsseelastoxfordsep
glossaries-extra v1.47+
Used instead of \glsseelastsep if the list contains three or more labels. This defaults to \glsseelastsep.

\glsseelastsep
glossaries
The separator used between the penultimate and ultimate entries of a cross-reference list.

\glsseelist{\langle \text{label list} \rangle}
glossaries
Iterates through the comma-separated list of entry labels to produce a formatted list, where each item in the list is encapsulated with \glsseeitem and each element is separated with \glsseesep or \glsseelastsep. This command was provided for the use of \glsseeformat to format cross-reference lists but may be used for any list of entry labels. This command is redefined by glossaries-extra (v1.47+) to additionally use \glsseefirstitem and \glsseelastoxfordsep.

\glsseesep
glossaries
The separator used between all but the last entries of a cross-reference list.

\glssetcategoryattribute{\langle \text{category} \rangle}{\langle \text{attribute} \rangle}{\langle \text{value} \rangle}
glossaries-extra
Sets the value of the attribute for the given category.

\glssetexpandfield{\langle \text{field} \rangle}
glossaries
Switches on field expansion for the given field.

\glssetnoexpandfield{\langle \text{field} \rangle}
glossaries
Switches off field expansion for the given field.

\glssetwidest{\langle \text{level} \rangle}{\langle \text{text} \rangle}
glossary-tree
Used with the alttree style to set the widest entry name for the given level.
General Command Summary

```latex
\glsshortaccessdisplay{(text)}{(label)} \quad \text{glossaries-access}
Displays \langle text \rangle with the accessibility support provided by \glsentryshortaccess{(label)}.\n
\glsshortpluralaccessdisplay{(text)}{(label)} \quad \text{glossaries-access}
Displays \langle text \rangle with the accessibility support provided by \glsentryshortpluralaccess{(label)}.\n
\glsshorttok\quad \text{glossaries}
Token register used in the construction of acronyms or abbreviations to allow the style hooks to access the short form.\n
\glsshowtarget{(label)} \quad \text{glossaries v4.32+}
Used to show the target name when the \texttt{debug=\{showtargets\}} option is on.\n
\glsstartrange{\langle options\rangle}{\langle label list\rangle} \quad \text{glossaries-extra v1.50+}
Essentially like \texttt{\glsaddeach[format=\{\},\langle options\rangle]}{\langle label-list\rangle}. If \texttt{format} is used in \langle \texttt{options} \rangle, the open marker ( will be inserted in front of the value.\n
\glssstepentry{(label)} \quad \text{glossaries}
Increments the glossaryentry counter, which is defined with the \texttt{entrycounter} option, and automatically labels it with \texttt{\label}.\n
\glssubentrycounterlabel \quad \text{glossaries}
Govern the way the glossarysubentry counter is displayed by \texttt{\glssubentryitem}.\n
\glssubentryitem{(label)} \quad \text{glossaries v3.0+}
Increments and displays the glossarysubentry counter, if appropriate.\n
\glssubgroupheading{\langle previous level\rangle}{\langle level\rangle}{\langle parent-label\rangle}{\langle group-label\rangle} \quad \text{glossaries-extra v1.49+}
Formats the heading for the sub-group identified by the given label.\n
\glssymbol{\langle options\rangle}{\langle label\rangle}{\langle insert\rangle} \quad \text{glossaries}
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{symbol} field without altering the first use flag.\n
\glssymbolaccessdisplay{(text)}{(label)} \quad \text{glossaries-access}
Displays \langle text \rangle with the accessibility support provided by \glsentrysymbolaccess{(label)}.\n
\glssymbolpluralaccessdisplay{(text)}{(label)} \quad \text{glossaries-access}
Displays \langle text \rangle with the accessibility support provided by \glsentrysymbolpluralaccess{(label)}.\n
\glssymbolsgroupname \quad \text{glossaries}
Language-sensitive name used for the symbols group and also used for the title of the glossary created with the \texttt{symbols} package option.\n
\glstarget{(label)}{(text)} \quad \text{glossaries v1.18+}
Creates a hypertarget for the entry given by \langle label \rangle (the target for commands like \texttt{\gls}) and displays \langle text \rangle.\n
604
\texttt{\glstext{(options)}{(label)}{(insert)}} \hspace{1cm} \text{glossaries}

As \texttt{\glstext} but converts the first letter to upper case.

\texttt{\glstext{(options)}{(label)}{(insert)}} \hspace{1cm} \text{glossaries}

Links to the entry’s definition in the glossary with the link text obtained from the \texttt{text} field without altering the first use flag.

\texttt{\glstextaccessdisplay{(text)}{(label)}} \hspace{1cm} \text{glossaries-access}

Displays \texttt{(text)} with the accessibility support provided by \texttt{\glsentrytextaccess{(label)}}.

\texttt{\glstextformat{(text)}} \hspace{1cm} \text{glossaries}

Used by commands like \texttt{\textsc{glsl}} to format the link text.

\texttt{\glstextup{(text)}} \hspace{1cm} \text{glossaries v3.09a+}

Typesets \texttt{(text)} in an upright font (used to cancel the effect of \texttt{\textsc{textsc}} in abbreviation styles that use \texttt{\glisabbrvscfont}).

\texttt{\glstildechar} \hspace{1cm} \text{glossaries*}

Expands to a literal tilde \texttt{~} character.

\texttt{\glstreedefaultnamefmt{(text)}} \hspace{1cm} \text{glossaries-extra-stylemods v1.31+}

Used as the default format for \texttt{\glstreenamefmt}, \texttt{\glstreegroupphonenumberfmt} and \texttt{\glstreenavigationfmt}.

\texttt{\glstreegroupphonenumberfmt{(text)}} \hspace{1cm} \text{glossary-tree v4.22+ and glossaries-extra-stylemods v1.31+}

Used with the tree styles to format the group headings.

\texttt{\glstreenamefmt{(text)}} \hspace{1cm} \text{glossary-tree v4.08+ and glossaries-extra-stylemods v1.31+}

Used with the tree styles to format the entry’s name.

\texttt{\glstreenavigationfmt{(text)}} \hspace{1cm} \text{glossary-tree v4.22+ and glossaries-extra-stylemods v1.31+}

Used with the tree styles to format the navigation elements.

\texttt{\glstreennonamedesc{(label)}} \hspace{1cm} \text{glossaries-extra-stylemods v1.31+}

Displays the pre-description separator, the description and the post-description hook for the treenoname styles.

\texttt{\glstreepredesc{(label)}} \hspace{1cm} \text{glossary-tree v4.26+}

Separator used before the description for the tree styles.

\texttt{\glstreeprelocation} \hspace{1cm} \text{glossaries-extra-stylemods v1.21+}

Inserted before the location list for top-level entries in the tree-like styles.

\texttt{\glstriggerrecordformat{(text)}} \hspace{1cm} \text{glossaries-extra v1.21+}

Does nothing but when used as a location format \texttt{\bib2gls} recognises it as an ignored record indexed by commands like \texttt{\rgls}.

\texttt{\glsunset{(label)}} \hspace{1cm} \text{glossaries}

Unsets the first use flag so that the entry is marked as having been used.

\texttt{\glsupdatewidest{(level)}{(text)}} \hspace{1cm} \text{glossaries-extra-stylemods v1.23+}

As \texttt{\glissetwidest} but only sets if \texttt{(text)} is wider than the current value.
General Command Summary

\glssupercase{(text)} \hfill Glossaries v4.50+*
Converts \langle text \rangle to upper case.

\glsuseabbrvfont{(text)}{(category)} \hfill Glossaries-extra v1.21+
Applies the formatting command used for the short form for the abbreviation style
associated with the given category.

\glsuselongfont{(text)}{(category)} \hfill Glossaries-extra v1.21+
Applies the formatting command used for the long form for the abbreviation style
associated with the given category.

\glsuserdescription{(description)}{(label)} \hfill Glossaries-extra v1.30+
Used with “user” abbreviation styles to encapsulate the description. Just does
\glslonguserfont{(description)} by default.

\glsuseri[(options)]{(label)}{(insert)} \hfill Glossaries
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{user1}
field without altering the first use flag.

\glsuserii[(options)]{(label)}{(insert)} \hfill Glossaries
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{user2}
field without altering the first use flag.

\glsuseriii[(options)]{(label)}{(insert)} \hfill Glossaries
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{user3}
field without altering the first use flag.

\glsuseriv[(options)]{(label)}{(insert)} \hfill Glossaries
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{user4}
field without altering the first use flag.

\glsuserv[(options)]{(label)}{(insert)} \hfill Glossaries
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{user5}
field without altering the first use flag.

\glsuservi[(options)]{(label)}{(insert)} \hfill Glossaries
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{user6}
field without altering the first use flag.

\glsxtr@record{(label)}{(prefix)}{(counter)}{(format)}{(location)} \hfill Glossaries-extra v1.08+
This command is written to the .aux file each time an entry is indexed to provide
\texttt{bib2gls} with the record information.

\glsxtr@record@nameref{(label)}{(prefix)}{(counter)}{(format)}{(location)}{(title)}\hfill Glossaries-extra v1.37+
\langle href \rangle\langle hcounter \rangle
Used instead of \glsxtr@record when the \texttt{record=nameref} option is used.

\glsxtr@resource[(options)]{(filename)} \hfill Glossaries-extra v1.08+
This internal command is written to the .aux file by \glsxtr@resourcefile to provide
\texttt{bib2gls} with the resource information.
General Command Summary

\glxtr@wrglossarylocation\{\langle n \rangle\}\{\langle page \rangle\}
glossaries-extra-bib2gls v1.29+
This command simply expands to \langle n \rangle, the value of the wrglossary counter for the given page.

\glxtrabbreviationfont\{\langle text \rangle\}
glossaries-extra v1.30+
Used by commands like \gls to format the link text for (non-regular) abbreviations.

\glxtrabbrvfootnote\{\langle label \rangle\}\{\langle long form \rangle\}
glossaries-extra v1.07+
Used with the “footnote” abbreviation styles to do the footnote. The \langle label \rangle is ignored by default. The \langle long form \rangle includes the font changing command. This just does \footnote{\langle long form \rangle}.

\glxtrabbrvpluralsuffix
glossaries-extra*
The default suffix used to construct the plural for the short form of abbreviations. This just uses \glspimalsuffix. If you don’t want a plural suffix, you can use the noshortplural attribute.

\glxtrabbrvtype
glossaries-extra
Expands to the default glossary type when using \newabbreviation.

\glxtraddlabelprefix\{\langle prefix \rangle\}
glossaries-extra-bib2gls v1.37+
Appends \langle prefix \rangle to the prefix label list.

\glxtrAltTreePar
glossaries-extra-stylemods v1.05+
Used by the alttree styles to indicate a paragraph break that retains the hanging indent.

\glxtralttreeSymbolDescLocation\{\langle label \rangle\}\{\langle location list \rangle\}
glossaries-extra-stylemods v1.05+
Used by the alttree styles to format the symbol, description and location.

\glxtrapptocsvfield\{\langle entry label \rangle\}\{\langle field label \rangle\}\{\langle value \rangle\}
glossaries-extra v1.47+
Appends a comma followed by \langle value \rangle to the given field for the given entry, it that field has already been set, otherwise it sets the field to just \langle value \rangle (there’s no check for the existence of either the entry or the field).

\GlsXtrAutoAddOnFormat\{\langle label \rangle\}\{\langle format list \rangle\}\{\langle glsadd options \rangle\}
glossaries-extra v1.37+
Makes commands like \gls and \glslink (but not \glsadd) automatically insert \glsadd[\langle glsadd options \rangle]{\langle label \rangle} if the format (supplied in the optional argument of the invoking \gls, \glslink etc) matches any in the given comma-separated elements of \langle format list \rangle. The format isn’t automatically applied to the \glsadd options.

\glxtrautoindexassignsort\{\langle cs \rangle\}\{\langle label \rangle\}
glossaries-extra v1.16+
Assigns the sort value for \index when using auto-indexing.

\glxtrautoindexentry\{\langle label \rangle\}
glossaries-extra v1.16+
Used for the actual value in \index when using auto-indexing.

\GlsXtrBibTeXEntryAliases
glossaries-extra-bib2gls v1.29+
Expands to the set of common entry aliases for @bibtexentry.

\glxtrbookindexname\{\langle label \rangle\}
glossary-bookindex
Used with the bookindex style to format the entry’s name.
General Command Summary

\glsxtrbookindexprelocation{⟨label⟩}  
glossary-bookindex  
Used with the bookindex style before the location list.

\glsxtrclearlabelprefixes{⟨prefix⟩}  
glossaries-extra-bib2gls v1.37+  
Clears the prefix label list.

\glsxtrcombiningdiacriticrules  
glossaries-extra-bib2gls v1.27+  
Collation sub-rule for combining diacritic characters.

\glsxtrcontrolrules  
glossaries-extra-bib2gls v1.27+  
Collation sub-rule for control characters usually placed at the start of a rule in the “ignored characters” section (although there typically won’t be any control codes in sort fields).

\glsxtrcopytoglossary{⟨label⟩}{⟨type⟩}  
glossaries-extra v1.12+  
Copies the entry given by ⟨label⟩ to the glossary given by ⟨type⟩.

\GlsXtrDefaultResourceOptions  
glossaries-extra v1.40+  
Provides default options for \glsxtrresourcefile.

\glsxtrdetoklocation{⟨location⟩}  
glossaries-extra v1.21+  
May be used to detokenize problematic locations, but just does its argument by default.

\glsxtrdigitrules  
glossaries-extra-bib2gls v1.27+  
Collation sub-rule for digits from the basic Latin set (0, ..., 9) as well as their subscript and superscript variants.

\glsxtrdisplaylocnameref{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨title⟩}{⟨href⟩}  
{⟨hcounter⟩}{⟨file⟩}  
glossaries-extra-bib2gls v1.37+  
Used to display a nameref location in the location field (with a hyperlink, if enabled).

\glsxtrdisplaysupplloc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨src⟩}  
{⟨location⟩}  
glossaries-extra-bib2gls v1.36+  
Used to display an external location in the supplementary list (with a hyperlink, if enabled).

\GlsXtrDualBackLink{⟨text⟩}{⟨label⟩}  
glossaries-extra-bib2gls v1.30+  
Creates a hyperlink to the dual entry whose label is obtained from the field given by \GlsXtrDualField.

\GlsXtrDualField  
glossaries-extra-bib2gls v1.30+  
The field used to store the dual label. This defaults to dual but will need to be redefined if a different value is given by dual-field.

\glsxtremsuffix  
glossaries-extra  
The suffix used to construct the plural for the short form of abbreviations with the “em” styles. This defaults to \glsxtrabrrvpluralsuffix.

\GlsXtrEnableInitialTagging{⟨category list⟩}{⟨cs⟩}  
glossaries-extra  
 Defines the control sequence ⟨cs⟩ to be used with abbreviation tagging with the given categories.
\glsxtrenablerecordcount \glsxtrenablerecordcount glossaries-extra v1.21+
Redefines \texttt{\textit{gls etc}} to their \texttt{\textit{rgls}} counterpart.

\glsxtrendfor \glsxtrendfor glossaries-extra v1.24+
May be used within the handler macro of \texttt{\textit{\glsxtrforcsvfield}} to prematurely break the loop.

\glsxtrentryfmt \glsxtrentryfmt glossaries-extra v1.12+
Alternative to \texttt{\textit{\glsxtrfmt}} for use in section headings.

\glsxtretryparentname \glsxtretryparentname glossaries-extra v1.39+
Expands to the entry’s parent’s name.

\GlsXtrExpandedFmt \GlsXtrExpandedFmt glossaries-extra v1.30+
Fully expands \texttt{\textit{text}} and then does \texttt{\textit{cs}}{\texttt{\textit{expanded text}}}.

\glsxtrfielddolistloop \glsxtrfielddolistloop glossaries-extra v1.12+
Iterates over the items the given field, which contains an etoolbox internal list.

\glsxtrfieldforlistloop \glsxtrfieldforlistloop glossaries-extra v1.29+
Iterates over the items the given field, which contains an etoolbox internal list, using the given handler.

\glsxtrfieldformatcsvlist \glsxtrfieldformatcsvlist glossaries-extra v1.42+
Formats the items in the given field, which contains a comma-separated list, using the same handler as \texttt{\textit{DTLformatlist}}.

\glsxtrfieldformatlist \glsxtrfieldformatlist glossaries-extra v1.42+
Formats the items in the given field, which contains an etoolbox internal list, using the same handler as \texttt{\textit{DTLformatlist}}.

\glsxtrfieldifinlist \glsxtrfieldifinlist glossaries-extra v1.12+
Tests if the given item is in the given field that contains an etoolbox internal list.

\glsxtrfieldlistadd \glsxtrfieldlistadd glossaries-extra v1.12+
Adds the given item to the given field that contains an etoolbox internal list.

\glsxtrfieldxifinlist \glsxtrfieldxifinlist glossaries-extra v1.12+
Tests if the expansion of the given item is in the given field that contains an etoolbox internal list.

\glsxtrfmt \glsxtrfmt glossaries-extra v1.12+
Formats the given text according to the formatting command identified by the value of the field obtained from \texttt{\textit{\GlsXtrFmtField}}.

\glsxtrfmt* \glsxtrfmt* glossaries-extra v1.23+
Like \texttt{\textit{\glsxtrfmt}} but inserts extra material into the link text but outside of the formatting command.

\GlsXtrFmtDefaultOptions \GlsXtrFmtDefaultOptions glossaries-extra v1.12+
The default options used by \texttt{\textit{\glsxtrfmt}}.
General Command Summary

\glsxtrfmtdisplay{(cs-name)}{(text)}{(insert)}
glossaries-extra

Used by \glsxtrfmt to format the given \text{\textit{text}} where \text{\textit{cs-name}} is obtained from the field identified by \GlsXtrFmtField and \text{\textit{insert}} is empty for the unstarred \glsxtrfmt and the final optional argument of the starred version \glsxtrfmt*.

\GlsXtrFmtField
glossaries-extra v1.12+

Expands to the internal label of the field used to store the control sequence name for use with \glsxtrfmt.

\glsxtrfootnotename
glossaries-extra v1.25+

Hook for the “footnote” abbreviation styles that expands to the value that the \text{\textit{name}} field is assigned to when the abbreviation is defined with \newabbreviation (defaults to the short form).

\glsxtrforcsvfield{(label)}{(field)}{(handler)}
glossaries-extra v1.24+

Iterates over the comma-separated list in the given \text{\textit{field}} for the entry identified by \text{\textit{label}} and performs \text{\textit{handler}}\{\text{\textit{element}}\} on each element of the list, where \text{\textit{handler}} is a control sequence which takes a single argument.

\GlsXtrForeignText{(label)}{(text)}
glossaries-extra v1.32+

Encapsulates \text{\textit{text}} in \foreignlanguage where the language label is obtained from the locale tag given in the field identified by \GlsXtrForeignTextField.

\GlsXtrForeignTextField
glossaries-extra v1.32+

Used by \GlsXtrForeignText to identify the field containing the locale tag.

\GlsXtrForUnsetBufferedList{(cs)}
glossaries-extra v1.31+

Iterates over all the entry whose labels are stored in the buffer that was started with \GlsXtrStartUnsetBuffering and implements \text{\textit{cs}}\{(label)\} at each iteration.

\glsxtrfractionrules
glossaries-extra-bib2gls v1.27+

Collation sub-rule for vulgar fraction characters.

\glsxtrfull{(options)}{(label)}
glossaries-extra

Links to the entry’s definition in the glossary with the link text obtained from the \text{\textit{long}} and \text{\textit{short}} fields (using the appropriate abbreviation style) without altering the first use flag.

\glsxtrfullsep{(label)}
glossaries-extra

The separator used in the full format for the parenthetical abbreviation styles or for inline parenthetical styles. This just does a space by default.

\glsxtrGeneralLatinIIIRules
glossaries-extra-bib2gls v1.27+

Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIIRules but includes D/ð between D/d and E/e and ß treated as “sz”).

\glsxtrGeneralLatinIIRules
glossaries-extra-bib2gls v1.27+

Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIIRules but includes D/ð between D/d and E/e and ß treated as “ss”).
General Command Summary

\glsxtrGeneralLatinIrules  glossaries-extra-bib2gls v1.27+
Collation sub-rule for Latin characters (basic Latin set plus subscript and superscript Latin characters).

\glsxtrGeneralLatinIVrules glossaries-extra-bib2gls v1.27+
Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIrules but includes D/ð between D/d and E/e and Æ/æ treated as AE/ae, OE/oe, Ï/p treated as TH/th and ß treated as “ss”).

\glsxtrGeneralLatinVIIIrules glossaries-extra-bib2gls v1.27+
Collation sub-rule for Latin characters: as \glsxtrGeneralLatinIrules but includes AE/æ treated as A/a, OE/œ treated as OE/œ, Ï/p treated as TH/th, ß treated as “ss”, D/ð treated as D/d, Ø/ø treated as O/o and Ł/l treated as L/l.

\glsxtrGeneralLatinVIIrules glossaries-extra-bib2gls v1.27+
Collation sub-rule for Latin characters: as \glsxtrGeneralLatinIrules but includes Æ/æ between A/a and B/b, D/ð between D/d and E/e, Ẅ/ẅ (insular G) as G/g, Ù/ù between O/o and P/p, f (long S) equivalent to S/s, Ï/p between T/t and U/u and Ï/p (wynn) as W/w.

\glsxtrGeneralLatinVIrules glossaries-extra-bib2gls v1.27+
Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIrules but includes Æ/æ between A/a and B/b, D/ð between D/d and E/e and Ï/p treated as TH/th and ß treated as “sz”).

\glsxtrgenerall Aurules glossaries-extra-bib2gls v1.27+
Collation sub-rule for general punctuation characters.

\glsxtrglossentry\{\langle label\rangle\} glossaries-extra v1.21
Displays the given entry name including a hypertarget (if hyperref has been loaded) as the destination for commands like \gls.

\glsxtrglossentryother\{\langle header\rangle\}\{\langle label\rangle\}\{\langle field\rangle\} glossaries-extra v1.22+
Like \glsxtrglossentry but uses the value given in the supplied internal \langle field\rangle where \langle header\rangle is the code to use in the header (leave empty for default).

\glsxtrgroupfield glossaries-extra v1.21+
Expands to the field label used to store the entry group labels.

\GLSXTRhiernames glossaries-extra v1.37+
Displays the hierarchical name for the entry identified by \langle label\rangle with each level separated by \glsxtrhiernamesep where each name is converted to upper case.

\GLSXTRhiernames glossaries-extra v1.37+
Displays the hierarchical name for the entry identified by \langle label\rangle with each level separated by \glsxtrhiernamesep where the top-most name is converted to upper case.

\GlsXtrhiernames glossaries-extra v1.37+
Displays the hierarchical name for the entry identified by \langle label\rangle with each level
separated by \glsxtrhiernamesep where each name has the first letter converted to upper case.

\Glsxtrhiername{⟨label⟩}
glossaries-extra v1.37+
Displays the hierarchical name for the entry identified by ⟨label⟩ with each level separated by \glsxtrhiernamesep where the top-most name has the first letter converted to upper case.

\glsxtrhiernamesep
glossaries-extra v1.37+
Separator between hierarchical levels displayed with \glsxtrhiername (and case-changing variants). This defaults to “▷” with the glossaries-extra package, but the bib2gls interpreter has a different definition to assist sorting.

\glsxtrhyphenrules
glossaries-extra-bib2gls v1.27+
Collation sub-rule for hyphen characters.

\glsxtrhyphensuffix
glossaries-extra v1.17+
The suffix used to construct the plural for the short form of abbreviations with the “hyphen” styles.

\GlsXtrIfFieldCmpNum{⟨field⟩}{⟨entry label⟩}{⟨comparison⟩}{⟨number⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.31+
Compares the given (numerical) field value to the given integer ⟨number⟩. The ⟨comparison⟩ may be one of: =, < or >. If the field is undefined or empty, the value is assumed to be 0. If the field is set, it must expand to an integer value. The value can be referenced in ⟨true⟩ or ⟨false⟩ with \glscurrentfieldvalue. The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIfFieldEqNum{⟨field⟩}{⟨entry label⟩}{⟨number⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.31+
Tests if the given field value expands to the given integer ⟨number⟩. If the field is undefined or empty, the value is assumed to be 0. If the field is set, it must expand to an integer value. The value can be referenced in ⟨true⟩ or ⟨false⟩ with \glscurrentfieldvalue. The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIfFieldEqStr{⟨field label⟩}{⟨entry label⟩}{⟨text⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.21+
Tests if the given field value is the same as ⟨text⟩ for the given entry, which may not exist. The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIfFieldEqXpStr{⟨field label⟩}{⟨entry label⟩}{⟨text⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.31+
Like \GlsXtrIfFieldEqStr but first (protected) fully expands ⟨text⟩ (but not the field
value). The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIfFieldNonZero{⟨field⟩}{⟨entry label⟩}{⟨true⟩}{⟨false⟩} glossaries-extra v1.31+
Tests if the given field value expands to a non-zero integer. If the field is undefined or empty, the value is assumed to be 0. If the field is set, it must expand to an integer value. The value can be referenced in ⟨true⟩ or ⟨false⟩ with \glscurrentfieldvalue. The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIfFieldUndef{⟨field label⟩}{⟨entry label⟩}{⟨true⟩}{⟨false⟩} glossaries-extra v1.23+
Tests if the given field (identified by its internal field label) isn’t defined for the given entry, which may also not exist.

\glsxtrifhasfield{⟨field label⟩}{⟨entry label⟩}{⟨true⟩}{⟨false⟩} glossaries-extra v1.19+*
Tests if the given entry has the given internal field set (defined and not empty) without testing if the entry exists and adds implicit scoping to ⟨true⟩ and ⟨false⟩.

\glsxtrifhasfield*{⟨field label⟩}{⟨entry label⟩}{⟨true⟩}{⟨false⟩} glossaries-extra v1.19+*
Tests if the given entry has the given field set (defined and not empty) without testing if the entry exists and without introducing an implicit scope.

\GlsXtrIfUnusedOrUndefined{⟨label⟩}{⟨true⟩}{⟨false⟩} glossaries-extra v1.34+
Does ⟨true⟩ if the entry given by ⟨label⟩ hasn’t been used or is undefined, otherwise it does ⟨false⟩. This command is not for use in the post-link hooks.

\glsxtrifwasfirstuse{⟨true⟩}{⟨false⟩} glossaries-extra
Only for use in the post-link hooks this tests if the entry just referenced was used for the first time.
General Command Summary

\GlsXtrIfXpFieldEqXpStr{{\langle field label \rangle}}{{\langle entry label \rangle}}{{\langle text \rangle}}{{\langle true \rangle}}{{\langle false \rangle}}\glsxtrextra v1.31+

Like \GlsXtrIfFieldEqStr but first (protected) fully expands both the field value and \langle text \rangle. The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIndexCounterLink{{\langle text \rangle}}{{\langle label \rangle}}\glsxtrextrasbib2gls v1.29+

Creates a hyperlink to the wrglossary location obtained from the indexcounter field.

\glsxtrindexseealso{{\langle label \rangle}}{{\langle xr list \rangle}}\glsxtrextra v1.16+

Indexes a “see also” cross-reference.

\glsxtrinsertinsidefalse\glsxtrextra v1.02+

Sets the ifglsxtrinsertinside switch to false.

\glsxtrinsertinsidetrue\glsxtrextra v1.02+

Sets the ifglsxtrinsertinside switch to true.

\glsxtrLatinAA\glsxtrextrasbib2gls v1.27+

Collation sub-rule for Å/å.

\glsxtrLatinOslash\glsxtrextrasbib2gls v1.27+

Collation sub-rule for Ø/ø.

\GlsXtrLoadResources[{\langle options \rangle}]\glsxtrextra v1.11+

A shortcut command that uses \glsxtrresourcefile.

\glsxtrlocalsetgrouptitle{{\langle group label \rangle}}{{\langle group title \rangle}}\glsxtrextra v1.24+

Locally sets the title for the group identified by the given label.

\GlsXtrLocationField\glsxtrextra v1.37+

Expands to the internal name of the field storing the location list, defaulting to location.

\glsxtrlocationhyperlink{{\langle counter \rangle}}{{\langle prefix \rangle}}{{\langle location \rangle}}\glsxtrextra v1.14+

Used to create the location hyperlink, this tests if an internal or external link is required depending on the definition of \glsxtrsupplocationurl.

\glsxtrlong[{\langle options \rangle}]{{\langle label \rangle}}\glsxtrextra

Links to the entry’s definition in the glossary with the link text obtained from the long field (using the appropriate abbreviation style) without altering the first use flag.

\glsxtrlonghyphen{{\langle long \rangle}}{{\langle label \rangle}}{{\langle insert \rangle}}\glsxtrextra v1.17+

Used by the long-hyphen-postshort-hyphen abbreviation to format the long form and check if the \langle insert \rangle starts with a hyphen.

\glsxtrlonghyphennoshort{{\langle label \rangle}}{{\langle long \rangle}}{{\langle insert \rangle}}\glsxtrextra v1.17+

Used by the “long-hyphen-noshort” styles to format the first use form. This checks if the inserted material starts with a hyphen and makes the appropriate modifications.

\glsxtrlonghyphenshort{{\langle label \rangle}}{{\langle long \rangle}}{{\langle short \rangle}}{{\langle insert \rangle}}\glsxtrextra v1.17+

Used by the “long-hyphen-short-hyphen” abbreviation styles to format the full form.
\glsxtrlongnoshortdescname\glossaries-extra v1.25+  
Hook for the long–noshort–desc abbreviation styles that expands to the value that the \texttt{name} field is assigned to when the abbreviation is defined with \texttt{\newabbreviation} (defaults to the long form).

\glsxtrlongnoshortname\glossaries-extra v1.25+  
Hook for the long–noshort abbreviation styles that expands to the value that the \texttt{name} field is assigned to when the abbreviation is defined with \texttt{\newabbreviation} (defaults to the short form).

\glsxtrlongshortdescname\glossaries-extra v1.17+  
Hook for the long–short–desc abbreviation styles that expands to the value that the \texttt{name} field is assigned to when the abbreviation is defined with \texttt{\newabbreviation} (defaults to the long form followed by the short form in parentheses).

\glsxtrlongshortname\glossaries-extra v1.25+  
Hook for the long–short abbreviation styles that expands to the value that the \texttt{name} field is assigned to when the abbreviation is defined with \texttt{\newabbreviation} (defaults to the short form).

\glsxtrlongshortuserdescname\glossaries-extra v1.25+  
Hook for the long–short–user–desc abbreviation styles that expands to the value that the \texttt{name} field is assigned to when the abbreviation is defined with \texttt{\newabbreviation} (defaults to the long form followed by the parenthetical material).

\glsxtrMathItalicGreekIrules\glossaries-extra-bib2gls v1.27+  
Collation sub-rule for math-Greek characters (includes upright digamma between epsilon and zeta).

\GLSxtrmultientryadjustedname{⟨\texttt{\sublist1}\⟩}{⟨\texttt{name}\⟩}{⟨\texttt{\sublist2}\⟩}{⟨\texttt{\label}\⟩}  
\texttt{\GLSxtrmultientryadjustedname}\texttt{\glossaries-extra-bib2gls v1.48+}  
Upper case version of \texttt{\glsxtrmultientryadjustedname}.

\GlsXtrmultientryadjustedname{⟨\texttt{\sublist1}\⟩}{⟨\texttt{name}\⟩}{⟨\texttt{\sublist2}\⟩}{⟨\texttt{\label}\⟩}  
\texttt{\GlsXtrmultientryadjustedname}\texttt{\glossaries-extra-bib2gls v1.48+}  
Title case version of \texttt{\glsxtrmultientryadjustedname}.

\Glsxtrmultientryadjustedname{⟨\texttt{\sublist1}\⟩}{⟨\texttt{name}\⟩}{⟨\texttt{\sublist2}\⟩}{⟨\texttt{\label}\⟩}  
\texttt{\Glsxtrmultientryadjustedname}\texttt{\glossaries-extra-bib2gls v1.48+}  
First letter uppercase version of \texttt{\glsxtrmultientryadjustedname}.

\glsxtrmultientryadjustedname{⟨\texttt{\sublist1}\⟩}{⟨\texttt{name}\⟩}{⟨\texttt{\sublist2}\⟩}{⟨\texttt{\label}\⟩}  
\texttt{\glsxtrmultientryadjustedname}\texttt{\glossaries-extra-bib2gls v1.48+}  
Used by \texttt{\compound-\adjust-name} to format the name using all the elements of the compound entry set, where \texttt{⟨\sublist1⟩} is the list of other labels before the main label, \texttt{⟨\sublist2⟩} is the list of other labels that follow the main label, \texttt{⟨name⟩} is the pre-adjustment name, and \texttt{⟨label⟩} identifies the compound entry.

\glsxtrmultisupplocation{⟨\texttt{\location}\⟩}{⟨\texttt{\src}\⟩}{⟨\texttt{\format}\⟩}  
\texttt{\glsxtrmultisupplocation}\texttt{\glossaries-extra-bib2gls v1.36+}  
Used by \texttt{\glsxtrdisplaysupploc} to format the external location (with a hyperlink, if enabled).
\glsxtrnewgls[⟨options⟩]{⟨prefix⟩}{⟨cs⟩}
glossaries-extra v1.21+
Defines the command ⟨cs⟩ to behave like \gls with the given label prefix.

\glsxtrnewglslike[⟨options⟩]{⟨prefix⟩}{⟨gls-like cs⟩}{⟨glspl-like cs⟩}{⟨Gls-like cs⟩}{⟨Glspl-like cs⟩}
glossaries-extra v1.21+
Defines commands to behave like \gls, \glspl, \Gls and \Glspl with the given label prefix.

\glsxtrnewnumber[⟨key=value list⟩]{⟨label⟩}
glossaries-extra numbers
Defines a new number.

\glsxtrnewsymbol[⟨key=value list⟩]{⟨label⟩}{⟨symbol⟩}
glossaries-extra symbols
Defines a new symbol.

\glsxtrnonprintablerules
glossaries-extra-bib2gls v1.27+
Collation sub-rule for non-printable characters.

\glsxtrnopostpunc
glossaries-extra v1.22+
Suppresses the post-description punctuation without suppressing the post-description hook.

\glsxtronlydescname
glossaries-extra v1.17+
Hook for the long-only–short-only–desc style that expands to the value that the name field is assigned to when the abbreviation is defined with \newabbreviation (defaults to the long form).

\glsxtronlyname
glossaries-extra v1.25+
Hook for the long-only–short-only style that expands to the value that the name field is assigned to when the abbreviation is defined with \newabbreviation (defaults to the short form).

\glsxtronlysuffix
glossaries-extra v1.17+
The suffix used to construct the plural for the short form of abbreviations with the “only” styles. The default definition just uses \glsxtrabbrvpluralsuffix.

\glsxtrp{⟨field⟩}{⟨label⟩}
glossaries-extra v1.07+
Displays the given ⟨field⟩ value for the entry given by ⟨label⟩ (no hyperlinks, except in the glossary, and no indexing by default, but includes formatting, if appropriate).

\glsxtrpageref{⟨label⟩}
glossaries-extra v1.11
When used with entrycounter or subentrycounter may be used to cross-reference the entry’s number in the glossary list with \pageref.

\glsxtrparen{⟨text⟩}
glossaries-extra v1.17+
Used to markup parenthetical material, such as in \glsxtrpostlinkAddDescOnFirstUse or in the long–short and short–long abbreviation styles.

\glsxtrpostdescabbreviation
glossaries-extra
Hook used after the description is displayed in the glossary for entries that have the category set to abbreviation.
\glsxtrpostdesc\texttt{(category)} \texttt{glossaries-extra}

Hook used after the description is displayed in the glossary for entries that have the 
\texttt{category} set to \texttt{(category)}. Common category hooks such as 
\glsxtrpostdesc\texttt{general} are provided by \texttt{glossaries-extra}. If required, this hook can be 
defined with \glsdefpostdesc.

\glsxtrpostdesc\texttt{general} \texttt{glossaries-extra}

Hook used after the description is displayed in the glossary for entries that have the 
\texttt{category} set to \texttt{general}.

\glsxtrpostdesc\texttt{symbol} \texttt{glossaries-extra}

Hook used after the description is displayed in the glossary for entries that have the 
\texttt{category} set to \texttt{symbol}.

\glsxtrposthyphenlong\texttt{\{label\}\{insert\}} \texttt{glossaries-extra v1.17+}

Used by the “postlong-hyphen” styles in the post-link hook.

\glsxtrposthyphenshort\texttt{\{label\}\{insert\}} \texttt{glossaries-extra v1.17+}

Used by the long-hyphen-postshort-hyphen style in the post-link hook.

\glsxtrposthyphensubsequent\texttt{\{label\}\{insert\}} \texttt{glossaries-extra v1.17+}

Used by the long-hyphen-postshort-hyphen abbreviation in the post-link hook for 
subsequent use.

\glsxtrpostlinkAddDescOnFirstUse \texttt{glossaries-extra}

Only for use in the post-link hooks, this appends a space and the value of the 
\texttt{description} field in parentheses if the entry that was just referenced was used for the 
first time.

\glsxtrpostlinkAddSymbolDescOnFirstUse \texttt{glossaries-extra v1.31+}

Only for use in the post-link hooks, if the entry that was just referenced was used for the 
first time, this appends a space and, in parentheses, the value of the \texttt{symbol} field (if set) 
followed by the value of the \texttt{description} field.

\glsxtrpostlinkAddSymbolOnFirstUse \texttt{glossaries-extra}

Only for use in the post-link hooks, this appends a space and the value of the \texttt{symbol} 
field in parentheses if the entry that was just referenced was used for the first time and 
has the \texttt{symbol} field set.

\glsxtrpostlink\texttt{(category)} \texttt{glossaries-extra}

Hook used after commands like \texttt{\gls} for entries that have the \texttt{category} set to 
\texttt{(category)}. If required, this hook can be defined with \glsdefpostlink.

\glsxtrpostname\texttt{(category)} \texttt{glossaries-extra}

Hook used by \texttt{\glossentryname} for entries that have the \texttt{category} set to \texttt{(category)}. If 
required, this hook can be defined with \glsdefpostname.

\glsxtrprelocation \texttt{glossary-bookindex v1.21+ and glossaries-extra-stylemods v1.21+}

Used before the location list in the bookindex style and the styles patched by 
glossaries-extra-stylemods.
\glsxtrprependlabelprefix\{prefix\}
glossaries-extra-bib2gls v1.37+
Prepends \textit{prefix} to the prefix label list.

\GlsXtrProvideBibTeXFields
\textit{\glsxtrprependlabelprefix}\{\textit{prefix}\}\textit{\glsaddstoragekey}
Define the standard \textit{\LaTeX} fields using \glsxtrprovidecommand\{\textit{cs}\}\{\textit{n}\}\{\textit{def}\}\{\textit{code}\}
Behaves like \providecommand in the document but like \textit{\renewcommand} in \textit{\bib2gls}.
\glsxtrprovidecommand\{\textit{key}\}\{\textit{default value}\}\{\textit{no link cs}\}
Add a new key, if not already defined, for use in \textit{\newglossaryentry} and an associated
\GlsXtrResourceInitEscSequences
Locally redefines quark commands, such as \textit{\u} and \textit{\textbackslash NULL}, that shouldn’t expand in
resource options as they have special meanings for some options. May be added to the
definition of \glsxtrresourceinit if required.
\glsxtrscsuffix
The suffix used to construct the plural for the short form of abbreviations with the
small-cap “sc” styles. This counteracts the effect of \textit{\textsc} using \textit{\glstextup}.
\glsxtrseelist\{\textit{xr label list}\}
Formats the list of cross-reference labels, without the initial “see” tag.
\glsxtrsetaliasnoindex
Hooks into the alias \textit{\noindex} setting.
\GlsXtrSetDefaultGlsOpts\{\textit{options}\}
Set the default options for commands like \textit{\gls}.
\GlsXtrSetDefaultNumberFormat\{\textit{format}\}
Set the default format to use if the \textit{format} key isn’t set.
Assigns the given \langle value \rangle to the field identified by \langle field label \rangle for the entry identified by \langle entry label \rangle.

\glsxtrsetglossarylabel{\langle label \rangle}
glossaries-extra v1.39+
Sets the label for subsequent glossaries (should be scoped or updated per glossary to prevent duplicate labels) and defines \@currentlabelname to the glossary’s TOC title. This is an alternative to the numberedsection=\{nameref\} package option or label \printunsrtglossary option.

\glsxtrsetgrouptitle{\langle group label \rangle}{\langle group title \rangle}
glossaries-extra v1.14+
Globally sets the title for the group identified by the given label.

\glsxtrsetpopts{\langle options \rangle}
glossaries-extra v1.07+
Sets the default options for \glsxtrp.

\GlsXtrSetRecordCountAttribute{\langle category list \rangle}{\langle value \rangle}
glossaries-extra v1.21+
Sets the recordcount attribute to \langle value \rangle for the given categories.

\glsxtrshort{\langle options \rangle}{\langle label \rangle}
glossaries-extra
Links to the entry’s definition in the glossary with the link text obtained from the short field (using the appropriate abbreviation style) without altering the first use flag.

\glsxtrshortdescname
Hook for the short–nolong–desc abbreviation styles that expands to the value that the name field is assigned to when the abbreviation is defined with \newabbreviation (defaults to the short form followed by long form in parentheses).

\glsxtrshorthyphen{\langle short \rangle}{\langle label \rangle}{\langle insert \rangle}
glossaries-extra v1.17+
Used by the “postlong-hyphen” styles to format the short form and check if \langle insert \rangle starts with a hyphen.

\glsxtrshorthyphenlong{\langle label \rangle}{\langle short \rangle}{\langle long \rangle}{\langle insert \rangle}
glossaries-extra v1.17+
Used by the short–hyphen–long–hyphen style to format the full form.

\glsxtrshortlongdescname
Hook for the short–long–desc abbreviation styles that expands to the value that the name field is assigned to when the abbreviation is defined with \newabbreviation (defaults to the long form followed by the short form in parentheses).

\glsxtrshortlongname
glossaries-extra v1.25+
Hook for the short–long abbreviation styles that expands to the value that the name field
is assigned to when the abbreviation is defined with `\newabbreviation` (defaults to the short form).

`\glsxtrshortlonguserdescname` \textit{glossaries-extra v1.25+}

Hook for the short–long–user–desc abbreviation styles that expands to the value that the `name` field is assigned to when the abbreviation is defined with `\newabbreviation` (defaults to the short form followed by the parenthetical material).

`\glsxtrshortnolongname` \textit{glossaries-extra v1.25+}

Hook for the short–nolong abbreviation styles that expands to the value that the `name` field is assigned to when the abbreviation is defined with `\newabbreviation` (defaults to the short form).

`\glsxtrsmsuffix` \textit{glossaries-extra}

The suffix used to construct the plural for the short form of abbreviations with the “sm” styles. This defaults to `\glsxtrabbvpluralsuffix`.

`\glsxtrspacerules` \textit{glossaries-extra-bib2gls v1.27+}

Collation sub-rule for space characters.

`\GlsXtrStandaloneEntryName{⟨label⟩}` \textit{glossaries-extra v1.37+}

Used within `\glsxtrglossentry` to display the name (with a hypertarget, if supported).

`\GlsXtrStandaloneEntryOther{⟨label⟩}{⟨field⟩}` \textit{glossaries-extra v1.37+}

Used within `\glsxtrglossentryother` to display the given field value (with a hypertarget, if supported).

`\GlsXtrStandaloneGlossaryType` \textit{glossaries-extra v1.31+}

Expands to the label for `\currentglossary` within `\glsxtrglossentry` and `\glsxtrglossentryother`.

`\GlsXtrStandaloneSubEntryItem{⟨label⟩}` \textit{glossaries-extra v1.31+}

Used within `\glsxtrglossentry` and `\glsxtrglossentryother` to display sub-item labels.

`\GlsXtrStartUnsetBuffering` \textit{glossaries-extra v1.30+}

Starts buffering calls to `\glsunset` (which is internally used by commands like `\gls`) for use in code where the boolean switch causes a problem. The buffer can later be processed and cleared with `\GlsXtrStopUnsetBuffering`. The starred form (added to v1.31) avoids duplicate labels in the buffer’s internal list.

`\GlsXtrStopUnsetBuffering` \textit{glossaries-extra v1.30+}

Unsets (locally with the starred form) the first use flag of all the entry whose labels are stored in the buffer that was started with `\GlsXtrStartUnsetBuffering` and then clears the buffer.

`\glsxtrsupphypernumber{⟨location⟩}` \textit{glossaries-extra v1.14+}

Uses `\glshypernumber` to create a hyperlink to the given location (if hyperlinks are supported) but first checks the `externallocation` attribute to determine if an external link is required.
General Command Summary

\glsxtrsuppplocationurl \quad \text{glossaries-extra v1.14+}
Set by \glsxtrsupphypernumber and \glsxtrmultisupplocation to the URL of the supplemental document for use by \glshypernumber.

\glsxtrtagfont\{\text\} \quad \text{glossaries-extra}
Font used by tagging command defined by \GlsXtrEnableInitialTagging.

\glsxtrunsrtdo\{\label\} \quad \text{glossaries-extra v1.12+}
Displays the entry given by \langle \label \rangle using \glossentry or \subglossentry depending on the entry’s hierarchical level (taking leveloffset into account).

\GLSxtrusefield\{\langle \text \rangle}\{\langle \label \rangle\} \quad \text{glossaries-extra v1.12+}
As \glsxtrusefield but converts the value to upper case.

\Glsxtrusefield\{\langle \text \rangle}\{\langle \label \rangle\} \quad \text{glossaries-extra v1.12+}
Like \glsxtrusefield but converts the first letter to upper case.

\glsxtrusefield\{\langle \text \rangle\}{\langle \label \rangle}\quad \text{glossaries-extra v1.12+}
Expands to the value of the given field for the given entry.

\glsxtruserfield \quad \text{glossaries-extra v1.04+}
Used by the parenthetical abbreviation styles, this expands to the internal label of the field used to store the additional parenthetical material. The default value is \textit{useri}.

\glsxtruserparen\{\langle \text \rangle\}{\langle \label \rangle}\quad \text{glossaries-extra v1.04+}
Used by the “user” abbreviation styles to format the parenthetical material where \langle \text \rangle is the default parenthetical text and \langle \label \rangle is the entry’s label. This checks the field given by \glsxtruserfield and, if set, the \langle \text \rangle is followed by a comma and the user value.

\glsxtrusersuffix \quad \text{glossaries-extra v1.04+}
The suffix used to construct the plural for the short form of abbreviations with the “user” styles. The default definition just uses \glsxtrabbrvpluralsuffix.

\glsxtrusesee\{\langle \label \rangle\} \quad \text{glossaries-extra v1.06+}
Applies \glsxseeformat to the entry’s see field if not empty.

\glsxtruseseealso\{\langle \label \rangle\} \quad \text{glossaries-extra v1.16+}
Applies \glsxseeformat to the entry’s seealso field if not empty.

\glsxtruseseealsoformat\{\langle \text \rangle\} \quad \text{glossaries-extra v1.16+}
Used to format the entries whose labels are given in \langle \text \rangle as a list of “see also” cross-references.

\glsxtruseformat\{\langle \text \rangle\}{\langle \labels \rangle}\quad \text{glossaries}
Formats the entries identified in the comma separated list of labels as a set of cross-references.

\glsxtrword\{\langle \text \rangle\} \quad \text{glossaries-extra v1.17+}
Used to encapsulate each word in the long form of an abbreviation by the \textit{markwords} attribute.
Use: \texttt{\textbackslash glsxtrwordsep} \texttt{glossaries-extra v1.17+}

Used to mark spaces between each word in the long form of an abbreviation by the markwords attribute.

**H**

\texttt{\textbackslash heartsuit} \texttt{kernel command* (maths mode)}

Heart symbol (♥).

\texttt{\textbackslash hyperbf\{\langle text\rangle\}} \texttt{glossaries}

A location format that uses the bold font that also has a hyperlink (if enabled).

\texttt{\textbackslash hyperemph\{\langle text\rangle\}} \texttt{glossaries}

A location format that uses \texttt{\textbackslash emph} to set the font and also has a hyperlink (if enabled).

\texttt{\textbackslash hyperit\{\langle text\rangle\}} \texttt{glossaries}

A location format that uses the italic font that also has a hyperlink (if enabled).

\texttt{\textbackslash hyperlink\{\langle target name\rangle\}\{\langle text\rangle\}} \texttt{hyperref*}

Create a hyperlink to \texttt{\langle target name\rangle} with the given \texttt{\langle text\rangle}.

\texttt{\textbackslash hyperref} \texttt{hyperref}

This command has 2 forms:

\texttt{\textbackslash hyperref\{\langle URL\rangle\}\{\langle category\rangle\}\{\langle name\rangle\}\{\langle text\rangle\}}

Create a hyperlink to an external location with the anchor formed from \texttt{\langle category\rangle \langle name\rangle} and the displayed \texttt{\langle text\rangle}.

\texttt{\textbackslash hyperref\{\langle label\rangle\}\{\langle text\rangle\}}

Create an internal hyperlink with the displayed \texttt{\langle text\rangle} to the same place that \texttt{\textbackslash ref\{\langle label\rangle\}} would be linked. Note that the \texttt{\langle label\rangle} argument isn’t optional. The square bracket notation disambiguates from the syntax for the external form of \texttt{\textbackslash hyperref}.

\texttt{\textbackslash hyperrm\{\langle text\rangle\}} \texttt{glossaries}

A location format that uses the serif (Roman) font that also has a hyperlink (if enabled).

\texttt{\textbackslash hypersf\{\langle text\rangle\}} \texttt{glossaries}

A location format that uses the sans-serif font that also has a hyperlink (if enabled).

**I**

\texttt{\textbackslash ifcase\{\langle number\rangle\}} \texttt{\LaTeX primitive*}

Case conditional.

\texttt{\textbackslash ifcsdef\{\langle cs-name\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}} \texttt{etoolbox}

Tests if the control sequence given by \texttt{\langle cs-name\rangle} is defined.

\texttt{\textbackslash ifcsstrequal\{\langle cs-name1\rangle\}\{\langle cs-name2\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}} \texttt{etoolbox}

Tests if the replacement text of the command given by the control sequence name \texttt{\langle cs-name1\rangle} equals the replacement text of the command given by the control sequence name \texttt{\langle cs-name2\rangle}.

622
General Command Summary

\ifcsstring{⟨cs-name⟩}{⟨string⟩}{⟨true⟩}{⟨false⟩} etoolbox
Tests if the replacement text of the command given by the control sequence name ⟨cs-name⟩ equals ⟨string⟩.

\ifdef{⟨cs⟩}{⟨true⟩}{⟨false⟩} etoolbox*
Tests if the control sequence ⟨cs⟩ is defined.

\ifdefstrequal{⟨cs1⟩}{⟨cs2⟩}{⟨true⟩}{⟨false⟩} etoolbox
Tests if the replacement text of the command ⟨cs1⟩ equals the replacement text of the command ⟨cs2⟩.

\ifDTLlistskipempty datatool-base* v2.31+
Conditional that determines whether or not commands like \DTLformatlist should skip empty elements.

\IfFileExists{⟨file⟩}{⟨true⟩}{⟨false⟩} kernel command*
If the given ⟨file⟩ exists does ⟨true⟩ otherwise does ⟨false⟩.

\ifglossaryexists{⟨type⟩}{⟨true⟩}{⟨false⟩} glossaries
Tests if the glossary identified by ⟨type⟩ exists and does ⟨true⟩ if it does exists, otherwise does ⟨false⟩. The unstarred form treats ignored glossaries as non-existent.

\ifglossaryexists*{⟨type⟩}{⟨true⟩}{⟨false⟩} glossaries v4.46+ (or glossaries-extra v1.44+)
The starred form of \ifglossaryexists treats ignored glossaries as existing.

\ifglsentryexists{⟨label⟩}{⟨true⟩}{⟨false⟩} glossaries
Tests if the entry given by ⟨label⟩ exists.

\ifglsfieldcseq{⟨entry label⟩}{⟨field label⟩}{⟨cs-name⟩}{⟨true⟩}{⟨false⟩} glossaries v4.16+
Tests if the given entry has the given field value equal to the replacement text of the command given by the control sequence name ⟨cs-name⟩, where ⟨field label⟩ is the internal field label (not the key name). The test uses \ifcsstrequal.

\ifglsfielddefeq{⟨entry label⟩}{⟨field label⟩}{⟨cs⟩}{⟨true⟩}{⟨false⟩} glossaries v4.16+
Tests if the given entry has the given field value equal to the replacement text of the command given by ⟨cs⟩, where ⟨field label⟩ is the internal field label (not the key name). The test uses \ifdefstrequal.

\ifglsfielddeg{⟨entry label⟩}{⟨field label⟩}{⟨string⟩}{⟨true⟩}{⟨false⟩} glossaries v4.16+*
Tests if the given entry has the given field value equal to ⟨string⟩, where ⟨field label⟩ is the internal field label (not the key name). No expansion is performed in the test (which just uses \ifcsstring).

\ifglsfieldvoid{⟨field label⟩}{⟨entry label⟩}{⟨true⟩}{⟨false⟩} glossaries v4.50+*
Expands to ⟨true⟩ if the given entry doesn’t exist, or exists but doesn’t have the field (identified by its internal field label) defined or does have the field defined but the field is empty. Otherwise expands to ⟨false⟩. This is essentially like \GlsXtrIfFieldUndef but also tests for an empty value.

\ifglshaschildren{⟨entry label⟩}{⟨true⟩}{⟨false⟩} glossaries*
Tests if the given entry, which must be defined, has child entries. This method is inefficient as it has to iterate over all defined entries to determine which ones have ⟨entry
General Command Summary

\label as the value of the \texttt{parent} field. With \texttt{bib2gls}, a more efficient approach is to use \texttt{save-child-count} and test the value of the \texttt{childcount} field. The \LaTeX \ parser library recognises this command and will simply use the child count (regardless of whether or not the child count is saved).

\ifgls\has\desc\{\entry\label\}{\true}\{\false\} \glossaries*  
Tests if the given entry, which must be defined, has the \texttt{description} field set.

\ifgls\has\field\{\field\label\}\{\entry\label\}\{\true\}\{\false\} \glossaries*  
Tests if the given entry, which must be defined, has the \texttt{field} field set to a non-empty value. This is implemented in \texttt{bib2gls} in the same way as \texttt{\glsxtrif\has\field*}.

\ifgls\has\long\{\entry\label\}\{\true\}\{\false\} \glossaries*  
Tests if the given entry, which must be defined, has the \texttt{long} field set.

\ifgls\has\parent\{\entry\label\}\{\true\}\{\false\} \glossaries*  
Tests if the given entry, which must be defined, has the \texttt{parent} field set.

\ifgls\has\prefix\{\entry\label\}\{\true\}\{\false\} \glossaries-prefix  
Tests if the given entry, which must be defined, has the \texttt{prefix} field set to value that’s not empty.

\ifgls\has\prefix\first\{\entry\label\}\{\true\}\{\false\} \glossaries-prefix  
Tests if the given entry, which must be defined, has the \texttt{prefix\first} field set to value that’s not empty.

\ifgls\has\prefix\first\plural\{\entry\label\}\{\true\}\{\false\} \glossaries-prefix  
Tests if the given entry, which must be defined, has the \texttt{prefix\first\plural} field set to value that’s not empty.

\ifgls\has\prefix\plural\{\entry\label\}\{\true\}\{\false\} \glossaries-prefix  
Tests if the given entry, which must be defined, has the \texttt{prefix\plural} field set to value that’s not empty.

\ifgls\has\short\{\entry\label\}\{\true\}\{\false\} \glossaries*  
Tests if the given entry, which must be defined, has the \texttt{short} field set.

\ifgls\has\suppressed\desc\{\entry\label\}\{\true\}\{\false\} \glossaries  
Tests if the given entry, which must be defined, has the \texttt{description} field set to \texttt{\nopost\desc}.

\ifgls\has\symbol\{\entry\label\}\{\true\}\{\false\} \glossaries*  
Tests if the given entry, which must be defined, has the \texttt{symbol} field set to value that’s not empty and not \texttt{\relax}.

\ifgls\used\{\label\}\{\true\}\{\false\} \glossaries  
Does \texttt{(true)} if the entry given by \texttt{\label} has been used, \texttt{(false)} if the entry hasn’t been used and neither if the entry doesn’t exist (an error or warning message will occur and ?? will appear in the document). This command is not for use in the post-link hooks.
\texttt{\textbackslash ifglxsxtrinsertinside} \textit{glossaries-extra v1.02+}
Switch that determines whether or not inserted text (provided in the final optional argument of commands like \texttt{\textbackslash gls}) is inside or outside of the font changing commands in the predefined abbreviation styles. The default is \texttt{false}.

\texttt{\textbackslash ifignoredglossary\{\texttt{type}\}\{\texttt{true}\}\{\texttt{false}\}} \textit{glossaries v4.08+}
Tests if the glossary given by \texttt{\textbackslash{type}} was defined as an ignored glossary.

\texttt{\textbackslash ifnum\{\texttt{number1}\}\{\texttt{comparison}\}\{\texttt{number2}\}} \textit{\LaTeX{} primitive*}
Integer conditional.

\texttt{\textbackslash ifstrempty\{\texttt{string}\}\{\texttt{true}\}\{\texttt{false}\}} \textit{etoolbox}
Tests if \texttt{\textbackslash{string}} is empty.

\texttt{\textbackslash immediate\{file operation\}} \textit{\LaTeX{} primitive}
Perform the file operation immediately instead of the usual delay.

\texttt{\textbackslash IN \textit{bib2gls quark}}
A quark to denote “is a substring” conditional in assign-fields conditionals. This token needs to be protected from expansion in the argument of \texttt{\textbackslash GlsXtrLoadResources}.

\texttt{\textbackslash index\{\texttt{text}\}} \textit{kernel command}
Indexes the given term by writing the relevant information to an associated file that can then be processed by makeindex or xindy.

\texttt{\textbackslash indexname \textit{glossaries or language packages}}
Language-sensitive name used for the title of the glossary created with the \texttt{\textbackslash index} package option.

\texttt{\textbackslash input\{\texttt{file}\}} \textit{kernel command*}
Input the given file.

\texttt{\textbackslash INTERPRET\{element-list\}} \textit{bib2gls quark}
A quark to denote an interpreted element in assign-fields. This token needs to be protected from expansion in the argument of \texttt{\textbackslash GlsXtrLoadResources}.

\texttt{\textbackslash invfmt\{\texttt{maths}\}}
Example command.

\texttt{\textbackslash J \textit{primitive}}
The current job name, which is usually the name of the main .tex file without the extension.

\texttt{\textbackslash L \textit{kernel command*}}
Produces the upper case L-slash character Ł.
General Command Summary

\l \text{kernel command*}
Produces the lower case l-slash character ł.

\label\{\langle id \rangle\} \text{kernel command*}
Creates a label that can be referenced with \ref or \pageref.

\LABELIFY\{\langle element-list \rangle\} \text{bib2gls quark}
A quark to denote a label element in assign-fields. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources.

\LABELIFYLIST\{\langle element-list \rangle\} \text{bib2gls quark}
A quark to denote a label-list element in assign-fields. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources.

\LC\{\langle element-list \rangle\} \text{bib2gls quark}
A quark to denote a lower case change in assign-fields syntax. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources.

\LEN\{\langle element-list \rangle\} \text{bib2gls quark}
A quark to denote the number of (detokenized) characters in a value in assign-fields. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources.

\let\{\langle token1 \rangle\}\{\langle token2 \rangle\} \text{\LaTeX primitive*}
Assigns \langle token1 \rangle to \langle token2 \rangle.

\listbreak \text{etoolbox}
May be used within the handler macro of etoolbox’s internal list loop commands to prematurely break the loop.

\listxadd\{\langle list cs \rangle\}\{\langle element \rangle\} \text{etoolbox}
Globally adds (expanded) \langle element \rangle to the list stored in the control sequence \langle list cs \rangle.

\loadglsentries\{\langle type \rangle\}\{\langle file \rangle\} \text{glossaries}
Locally redefines \glsdefaulttype to \langle type \rangle and inputs \langle file \rangle.

\longnewglossaryentry\{\langle label \rangle\}\{\langle key=value list \rangle\}\{\langle description \rangle\} \text{glossaries}
Defines a new glossary entry and appends \leavemode\unskip\nopostdesc at the end of \langle description \rangle.

\longnewglossaryentry*\{\langle label \rangle\}\{\langle key=value list \rangle\}\{\langle description \rangle\} \text{glossaries-extra v1.12+}
Defines a new glossary entry without appending any extra code to the end of \langle description \rangle.

\longprovideglossaryentry\{\langle label \rangle\}\{\langle key=value list \rangle\}\{\langle description \rangle\} \text{glossaries}
Defines a new glossary entry if one doesn’t already exist with the given label.

\mainmatter \text{book-like classes}
Switches to main matter.

\footnotesize

626
\makefirstuc{\langle text \rangle}
  Converts the first letter of \langle text \rangle to upper case.

\makeglossaries
  Opens associated glossary files to be processed by makeindex or xindy.

\MakeLowercase{\langle text \rangle}
  kernel command*
  Converts \langle text \rangle to lower case.

\makenoidxglossaries
  glossaries v4.04+
  Indicates that \TeX should be used to sort and collate the glossary information instead of using an external application; this command should not be used with bib2gls.

\MakeTextLowercase{\langle text \rangle}
  textcase*
  Converts \langle text \rangle to lower case.

\MakeTextUppercase{\langle text \rangle}
  textcase*
  Converts \langle text \rangle to upper case.

\MakeUppercase{\langle text \rangle}
  kernel command*
  Converts \langle text \rangle to upper case.

\mathcal{\langle character \rangle}
  kernel command (maths mode)
  Renders the given (upper case) maths character in a calligraphic font.

\mathord{\langle maths \rangle}
  \TeX primitive
  Assigns the character or sub-formula in the argument to class 0, ordinary.

\MFUnocap{\langle word \rangle}
  mfirstuc* v1.09+
  Identifies \langle word \rangle as one that should not have its case-changed by \capitalisewords unless it occurs at the start.

\MFUskippunc{\langle punctuation \rangle}
  mfirstuc v2.07+*
  If \makefirstuc starts with a punctuation character it should be encapsulated with this command to skip \langle punctuation \rangle and apply the case-change to the following character.

\MFUwordbreak{\langle punctuation \rangle}
  mfirstuc v2.07+*
  If \capitalisewords contains punctuation that should be treated as a word break then \langle punctuation \rangle should be encapsulated with this command to apply the case-change to the following character.

\mgl\{\langle options \rangle\}\{\langle label \rangle\}\{\langle insert \rangle\}
  glossaries-extra v1.48+
  Applies \gl\ to each element in the set defined by \multiglossaryentry.

\MGP{\langle group-ref \rangle}
  bib2gls quark
  A quark to denote a reference to a group from a regular expression match. This token needs to be protected from expansion in the argument of \GlslXtrLoadResources. The \langle group-ref \rangle may be either an integer index or a textual name.

\midrule
  booktabs
  Horizontal rule for divider between header and main content of a tabular-like environment.
\texttt{\textbackslash mtxfmt\{(symbol)\}}

Example command.

\texttt{\textbackslash multiglossaryentry\{\langle\texttt{options}\rangle\}\{\langle\texttt{main label}\rangle\}\{\langle\texttt{list}\rangle\}}\texttt{\textbackslash glossaries-extra v1.48+}

Defines a set of labels (which must correspond to entries that have already been defined) that can be collectively referred to by commands like \texttt{\textbackslash mgls}. The \texttt{\langle\texttt{main label}\rangle} must be included in the comma-separated \texttt{\langle\texttt{list}\rangle} and indicates which element is considered the main entry in the set. If omitted, the last element in \texttt{\langle\texttt{list}\rangle} is assumed to be the main element.

\texttt{\textbackslash multiglossaryentrysetup\{\langle\texttt{options}\rangle\}}\texttt{\textbackslash glossaries-extra v1.48+}

Setup the general options for compound entries.

\texttt{\textbackslash n}

Indicates a newline character in regular expressions.

\texttt{\textbackslash nary\{(\texttt{text})\}}

Example command.

\texttt{\textbackslash newabbreviation\{\langle\texttt{key}=\texttt{value list}\rangle\}\{\langle\texttt{label}\rangle\}\{\langle\texttt{short}\rangle\}\{\langle\texttt{long}\rangle\}}\texttt{\textbackslash glossaries-extra}

Defines a new abbreviation.

\texttt{\textbackslash newacronym\{\langle\texttt{key}=\texttt{value list}\rangle\}\{\langle\texttt{label}\rangle\}\{\langle\texttt{short}\rangle\}\{\langle\texttt{long}\rangle\}}\texttt{\textbackslash glossaries}

Defines a new abbreviation. The \texttt{\textbackslash glossaries-extra} package redefines this to use \texttt{\textbackslash newabbreviation} with the \texttt{\texttt{category}} set to \texttt{\texttt{acronym}}.

\texttt{\textbackslash newcommand\{\langle\texttt{cs}\rangle\}\{\langle\texttt{n}\rangle\}\{\langle\texttt{def}\rangle\}\{\langle\texttt{code}\rangle\}}\texttt{\textbackslash kernel command^*}

Defines a new command.

\texttt{\textbackslash newdualentry\{\langle\texttt{key}=\texttt{value list}\rangle\}\{\langle\texttt{label}\rangle\}\{\langle\texttt{short}\rangle\}\{\langle\texttt{long}\rangle\}\{\langle\texttt{description}\rangle\}}

Example given in \texttt{\textbackslash glossaries} user manual.

\texttt{\textbackslash newentry\{\langle\texttt{label}\rangle\}\{\langle\texttt{key}=\texttt{value list}\rangle\}}\texttt{\textbackslash glossaries-extra \texttt{\texttt{shortcuts}}}

Equivalent to \texttt{\textbackslash newglossaryentry}.

\texttt{\textbackslash newglossary\{\langle\texttt{log}\rangle\}\{\langle\texttt{type}\rangle\}\{\langle\texttt{gls}\rangle\}\{\langle\texttt{glo}\rangle\}\{\langle\texttt{title}\rangle\}}\texttt{\textbackslash glossaries}

Defines a new glossary identified by \texttt{\langle\texttt{type}\rangle} with the given title and associated file extensions used by \texttt{makeindex} or \texttt{xindy}.

\texttt{\textbackslash newglossary*\{\langle\texttt{type}\rangle\}\{\langle\texttt{title}\rangle\}}\texttt{\textbackslash glossaries}

Defines a new glossary identified by \texttt{\langle\texttt{type}\rangle} with the given title.

\texttt{\textbackslash newglossaryentry\{\langle\texttt{label}\rangle\}\{\langle\texttt{key}=\texttt{value list}\rangle\}}\texttt{\textbackslash glossaries}

Defines a new glossary entry.

\texttt{\textbackslash newglossarystyle\{\langle\texttt{name}\rangle\}\{\langle\texttt{definition}\rangle\}}\texttt{\textbackslash glossaries}

Defines a new glossary style called \texttt{\langle\texttt{name}\rangle}.
\newignoredglossary\{⟨type⟩\}  
Defines a new ignored glossary (with hyperlinks suppressed) identified by ⟨type⟩ that’s not included in the list used by commands, such as \printunsrtglossaries, that iterate over defined glossaries.

\newignoredglossary*\{⟨type⟩\}  
Defines a new ignored glossary (without suppressing hyperlinks) identified by ⟨type⟩ that’s not included in the list used by commands, such as \printunsrtglossaries, that iterate over defined glossaries.

\newnum\{⟨label⟩\}\{⟨key=value list⟩\}  
Equivalent to \glsxtrnewnumber.

\newrobustcmd\{⟨cs⟩\}\[⟨n⟩]\[⟨def⟩\]{⟨code⟩}  
Behaves like \newcommand but the newly defined command will be robust.

\newsym\{⟨label⟩\}\{⟨key=value list⟩\}\{⟨symbol⟩\}  
Equivalent to \glsxtrnewsymbol.

\newterm\{⟨key=value list⟩\}\{⟨label⟩\}  
 Defines a new glossary entry where the description field defaults to empty.

\NG  
Produces the upper case eng I).

\ng  
Produces the lower case eng ŋ.

\NIN  
A quark to denote “is not a substring” conditional in assign-fields conditionals. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources.

\nobreakspace  
Produces a non-breakable space.

\NoCaseChange\{⟨text⟩\}  
Prevents \MakeTextUppercase and \MakeTextLowercase from converting ⟨text⟩.

\NOCHANGE\{⟨element-list⟩\}  
A quark to denote no case-change in assign-fields syntax. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources. It’s not defined by \GlsXtrResourceInitEscSequences.

\nopostdesc  
Suppresses the post-description hook.

\NOTPREFIXOF  
A quark to denote “is not a prefix of” conditional in assign-fields conditionals. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources.

\NOTSUFFIXOF  
A quark to denote “is not a suffix of” conditional in assign-fields conditionals. This token needs to be protected from expansion in the argument of \GlsXtrLoadResources.
General Command Summary

\NULL \textit{bib2gls quark}

A quark to denote a null value in \texttt{assign-fields} conditionals. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}.

\number{\textit{value}} \textit{\LaTeX primitive*}

Expands the given numerical \textit{value} to a base 10 integer number stripping any leading zeros (use \texttt{\char"\texttt{hex}} if the value is hexadecimal).

\numspacefmt{(\textit{symbol})} \textit{Example command.}

\textbf{O}

\O \textit{kernel command*}

Produces the upper case O-slash character Ø.

\o \textit{kernel command*}

Produces the lower case o-slash character ø.

\OE \textit{kernel command*}

Produces the upper case Œ-ligature.

\oe \textit{kernel command*}

Produces the lower case œ-ligature.

\oldacronym\{(\textit{label})\}{(\textit{short})}\{\{\textit{long}\}\}{\{\textit{options}\}\} \textit{glossaries}

Emulates the way the old \texttt{glossary} package defined acronyms.

\omicron \textit{glossaries-extra-bib2gls*}

Greek letter omicron ο.

\textbf{P}

\PackageError\{(\textit{name})\}{\{\textit{code}\}\{\textit{help}\}\} \textit{kernel command*}

Generates an error message for the package identified by \textit{name}.

\pagelistname \textit{glossaries}

Language-sensitive name used for the location list header for some glossary styles.

\pageref\{\{\textit{id}\}\} \textit{kernel command*}

Cross-reference the page where \texttt{\label\{\textit{id}\}} occurred.

\par \textit{kernel command*}

Paragraph break.

\parenswap\{(\textit{text1})\}{\{\textit{text2}\}\} \textit{Example command.}

\PGLS\{(\textit{options})\}{\{\textit{label}\}\}{\{\textit{insert}\}\} \textit{glossaries-prefix}

Does \texttt{(prefix)\GLS\{(\textit{options})\}{\{\textit{label}\}\}{\{\textit{insert}\}}} where the \texttt{\textit{prefix}} is obtained from the appropriate prefix field with the text converted to upper case.
General Command Summary

\texttt{\textbackslash pgls[\{options\}][\{label\}][\{insert\}]} \hspace{1cm} \texttt{\textbackslash glossaries-prefix}

Does \texttt{\textbackslash gls[\{options\}][\{label\}][\{insert\}]}, where the \texttt{\{prefix\}} is obtained from the appropriate prefix field with the first letter converted to upper case.

\texttt{\textbackslash pgls[\{options\}][\{label\}][\{insert\}]} \hspace{1cm} \texttt{\textbackslash glossaries-prefix}

Does \texttt{\textbackslash gls[\{options\}][\{label\}][\{insert\}]}, where the \texttt{\{prefix\}} is obtained from the appropriate prefix field.

\texttt{\textbackslash PGLSpl[\{options\}][\{label\}][\{insert\}]} \hspace{1cm} \texttt{\textbackslash glossaries-prefix}

Does \texttt{\textbackslash GLSpl[\{options\}][\{label\}][\{insert\}]}, where the \texttt{\{prefix\}} is obtained from the appropriate prefix field with the text converted to upper case.

\texttt{\textbackslash PgLspl[\{options\}][\{label\}][\{insert\}]} \hspace{1cm} \texttt{\textbackslash glossaries-prefix}

Does \texttt{\textbackslash GLSpl[\{options\}][\{label\}][\{insert\}]}, where the \texttt{\{prefix\}} is obtained from the appropriate prefix field with the first letter converted to upper case.

\texttt{\textbackslash pi} \hspace{1cm} \texttt{kern}\texttt{el command} \texttt{* (maths mode)}

Greek letter pi $\pi$.

\texttt{\texttt{\textbackslash PREFIXOF}} \hspace{1cm} \texttt{bib2gls quark}

A quark to denote “is a prefix of” conditional in \texttt{assign-fields} conditionals. This token needs to be protected from expansion in the argument of \texttt{\textbackslash Gl\texttt{s}XtrLoadResources}.

\texttt{\texttt{\textbackslash printglossaries}} \hspace{1cm} \texttt{\textbackslash glossaries}

Iterates over all non-ignored defined glossaries and performs \texttt{\textbackslash printglossary} for each one.

\texttt{\texttt{\textbackslash printglossary[\{options\}]}} \hspace{1cm} \texttt{\textbackslash glossaries}

Inputs file created by \texttt{makeindex} or \texttt{xindy}.

\texttt{\texttt{\textbackslash printnoidxglossaries}} \hspace{1cm} \texttt{\textbackslash glossaries v4.04+}

Iterates over all non-ignored defined glossaries and performs \texttt{\textbackslash printnoidxglossary} for each one.

\texttt{\texttt{\textbackslash printnoidxglossary[\{options\}]}} \hspace{1cm} \texttt{\textbackslash glossaries v4.04+}

Uses \textsf{Ti\TeX} to sort, collate and list the glossary.

\texttt{\texttt{\textbackslash printunsrtglossaries}} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.08+}

Iterates over all non-ignored defined glossaries and performs \texttt{\textbackslash printunsrtglossary} for each one.

\texttt{\texttt{\textbackslash printunsrtglossary[\{options\}]}} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.08+}

Display a glossary by iterating over all entries associated with that glossary in the order in which they were defined (which, with \texttt{bib2gls}, should correspond to the order obtained from the sort settings given in the resource set options).

Options:
entrycounter={⟨boolean⟩}
Locally enable or disable top-level enumeration (overrides entrycounter package option)

groups={⟨boolean⟩}
Controls whether or not \printunsrtglossary (or \printunsrtinnerglossary) should insert letter group markup. There’s no visible difference for glossary styles that don’t support letter groups (and nogroupskip is in effect) or if there’s no group information (for example, --no-group has been used)

label={⟨label⟩}
Creates a label for this glossary by locally using \glsxtrsetglossarylabel{⟨label⟩}

leveloffset={⟨n⟩}
Makes the glossary style act as though each entry’s hierarchical level is ⟨offset⟩ more than it actually is where ⟨offset⟩ is either ⟨n⟩ or is locally incremented by ⟨n⟩ if ⟨n⟩ starts with ++

nogroupskip={⟨boolean⟩}
Locally change whether or not to separate groups with a vertical space if the glossary style that support this option (overrides nogroupskip package option)

nonumberlist={⟨boolean⟩}
Locally change whether or not to display the location lists (overrides nonumberlist package option)

nopostdot={⟨boolean⟩}
Locally omit the post-description punctuation (overrides nopostdot and related package options)

numberedsection={⟨value⟩}
Locally change whether or not to use a numbered sectioning command (overrides numberedsection package option)

prefix={⟨label⟩}
Locally redefine \glolinkprefix for the item hypertargets and for any entry reference or cross-reference hyperlinks

style={⟨style-name⟩}
Use the glossary style identified by ⟨style-name⟩ (overrides current style setting)

subentrycounter={⟨boolean⟩}
Locally enable or disable level 1 enumeration (overrides subentrycounter package option)

target={⟨boolean⟩}
Locally enables or disables the hypertargets for each item

targetnameprefix={⟨label⟩}
Locally assign a prefix for the hypertargets assigned to each item (if target={true}) to avoid duplicate target names

title={⟨text⟩}
Locally sets the title for this glossary
General Command Summary

\texttt{toctitle={\{text\}}}  \\
Locally sets the TOC title for this glossary

\texttt{type={\{glossary-label\}}}  \\
Identifies the glossary list (default type, if omitted)

\texttt{\textbackslash printunsrtglossary*\{\{options\}\}\{\{code\}\}}}  \\
As \texttt{\textbackslash printunsrtglossary} but performs \texttt{\{code\}} first (scoped to localise any assignments within \texttt{\{code\}}).

\texttt{\textbackslash printunsrtglossaryentryprocesshook\{\{label\}\}}}  \\
Performed at each iteration of the internal loop used by \texttt{\textbackslash printunsrtglossary}.

\texttt{\textbackslash printunsrtglossaryhandler\{\{label\}\}}}  \\
Used by \texttt{\textbackslash printunsrtglossary} and \texttt{\textbackslash printunsrtinnerglossary} to handler each entry within the loop. By default this simply does \texttt{\textbackslash glsxtrunsrtdo}.

\texttt{\textbackslash printunsrtglossarypredoglossary}  \\
Hook performed by \texttt{\textbackslash printunsrtglossary}.

\texttt{\textbackslash printunsrtglossaryskipentry}  \\
Only allowed within \texttt{\textbackslash printunsrtglossaryentryprocesshook} this command indicates that the current entry should be skipped.

\texttt{\textbackslash printunsrtinnerglossary\{\{options\}\}\{\{pre code\}\}\{\{post code\}\}}}  \\
Similar to \texttt{\textbackslash printunsrtglossary} but only contains the code that would typically be placed inside the \texttt{theglossary} environment. This command should either be placed inside the \texttt{printunsrtglossarywrap} environment or inside the handler macro used by \texttt{\textbackslash printunsrtglossary}. This command is unsuitable for certain glossary styles, particularly tabular-like styles.

\texttt{\textbackslash ProcessOptions}  \\
Processes supplied options.

\texttt{\textbackslash protect\{token\}}  \\
Protects \texttt{\{token\}} from expansion.

\texttt{\textbackslash providecommand\{\{cs\}\}\{\{n\}\}\{\{def\}\}\{\{code\}\}}}  \\
Defines a command if it’s not already defined.

\texttt{\textbackslash provideglossaryentry\{\{label\}\}\{\{key=value list\}\}}}  \\
Defines a new glossary entry if one doesn’t already exist with the given label.

\texttt{\textbackslash provideignoredglossary\{\{type\}\}}}  \\
As \texttt{\textbackslash newignoredglossary} but does nothing if a glossary identified by \texttt{\{type\}} already exists.

\texttt{\textbackslash provideignoredglossary*\{\{type\}\}}}  \\
As \texttt{\textbackslash provideignoredglossary} but doesn’t suppress hyperlinks.

\texttt{\textbackslash providemultiglossaryentry\{\{options\}\}\{\{multilabel\}\}\{\{main label\}\}\{\{list\}\}}}  \\
As \texttt{\textbackslash multiglossaryentry} but does nothing if the label has already been defined as a compound entry.
General Command Summary

\ProvidesPackage{⟨name⟩}[⟨version⟩]
Identifies a package.

R
\ref{⟨id⟩}
Cross-reference the location where \label{⟨id⟩} occurred.
\refstepcounter{⟨counter name⟩}
Increments the given counter in a manner compatible with the \label cross-referencing mechanism.
\renewcommand{⟨cs⟩}[⟨n⟩][⟨def⟩]{⟨code⟩}
Redefines an existing command.
\RequirePackage[⟨options⟩]{⟨name⟩}[⟨min version⟩]
Loads the package identified by ⟨name⟩ from within another package.
\rgls[⟨options⟩]{⟨label⟩}[⟨insert⟩]
glossaries-extra v1.21+
Like \gls but checks for the record count trigger setting (the formatting is governed by \rglsformat).
\rglsformat{⟨label⟩}[⟨insert⟩]
glossaries-extra v1.21+
Used by \rgls if the record count switch is triggered.

S
\section[⟨toc title⟩]{⟨title⟩}
most classes that have a concept of document sections
Section heading.
\section*[⟨title⟩]
most classes that have a concept of document sections
Unnumbered section heading.
\seealso{⟨name⟩}
glossaries-extra
Language sensitive “see also” text (as from v1.42 this will be defined to \alsowname if that command exists).
\selectlanguage{⟨language name⟩}
babel and polyglossia
Switch to the rules of the given language.
\setabbreviationstyle[⟨category⟩]{⟨style-name⟩}
glossaries-extra
Sets the abbreviation style to ⟨style-name⟩ for the given ⟨category⟩, must be used before the abbreviation is defined.
\setcardfmt{⟨maths⟩}
Example command.
\setcontentsfmt{⟨contents⟩}
Example command.
\setentrycounter[⟨prefix⟩]{⟨counter⟩}
glossaries
Sets up the entry’s associated counter and prefix required by \glshypernumber.
\setfmt\{symbol\}
Example command.

\setglossarypreamble\{\langle type\rangle\}\{\langle code\rangle\}
glossaries
Sets \langle code\rangle as the preamble for the given glossary (or the default of \langle type\rangle is omitted).

\setglossarystyle\{\langle name\rangle\}
glossaries
Sets the glossary style identified by \langle name\rangle.

\setmainlanguage\{\langle options\rangle\}\{\langle language name\rangle\}
polyglossia
Load the main document language.

\setmembershiptfmt\{\langle variable(s)\rangle\}\{\langle condition\rangle\}
Example command.

\setmembershiptoneargfmt\{\langle variable(s)\rangle\}\{\langle condition\rangle\}
Example command.

\setotherlanguage\{\langle options\rangle\}\{\langle language name\rangle\}
polyglossia
Load a secondary document language.

\setupglossaries\{\langle key=\langle value list\rangle\}\}
glossaries
Applies the base glossaries options that are allowed to be changed after the package has loaded.

\showglocounter\{\langle label\rangle\}
glossaries
Interrupts the document build and shows the value of the counter field in the transcript.

\showglodesc\{\langle label\rangle\}
glossaries
Interrupts the document build and shows the value of the description field in the transcript.

\showglodescplural\{\langle label\rangle\}
glossaries
Interrupts the document build and shows the value of the descriptionplural field in the transcript.

\showglofield\{\langle entry label\rangle\}\{\langle internal field\rangle\}
glossaries
Interrupts the document build and shows the value of the given field in the transcript.

\showglofirst\{\langle label\rangle\}
glossaries
Interrupts the document build and shows the value of the first field in the transcript.

\showglofirstpl\{\langle label\rangle\}
glossaries
Interrupts the document build and shows the value of the firstplural field in the transcript.

\showgloflag\{\langle label\rangle\}
glossaries
Interrupts the document build and shows the value of the first use flag in the transcript.

\showgloglossaries
glossaries
Interrupts the document build and shows the list of all non-ignored glossary types in the transcript.
\showglolevel{\langle label \rangle} \glossaries
Interrupts the document build and shows the entry’s hierarchical level in the transcript.

\showgloloclist{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{loclist} field in the transcript.

\showglolong{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{long} field in the transcript.

\showgloname{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{name} field in the transcript.

\showgloparent{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{parent} field in the transcript.

\showgloplural{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{plural} field in the transcript.

\showgloshort{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{short} field in the transcript.

\showglosort{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{sort} field in the transcript.

\showglossarycounter{\langle type \rangle} \glossaries
Interrupts the document build and shows the default counter for the given glossary in the transcript.

\showglossaryentries{\langle type \rangle} \glossaries
Interrupts the document build and shows the list of entry labels for the given glossary in the transcript.

\showglossarytitle{\langle type \rangle} \glossaries
Interrupts the document build and shows the title of the given glossary in the transcript.

\showglosymbol{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{symbol} field in the transcript.

\showglosymbolplural{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{symbolplural} field in the transcript.

\showglotext{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{text} field in the transcript.

\showgloctype{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{type} field in the transcript.

\showglouser{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{user1} field in the transcript.

\showglouserii{\langle label \rangle} \glossaries
Interrupts the document build and shows the value of the \texttt{user2} field in the transcript.
General Command Summary

\showglouserii\{\textit{label}\}\textsuperscript{glossaries}
Interrupts the document build and shows the value of the user3 field in the transcript.

\showglouseriv\{\textit{label}\}\textsuperscript{glossaries}
Interrupts the document build and shows the value of the user4 field in the transcript.

\showgloserv\{\textit{label}\}\textsuperscript{glossaries}
Interrupts the document build and shows the value of the user5 field in the transcript.

\showgloservi\{\textit{label}\}\textsuperscript{glossaries}
Interrupts the document build and shows the value of the user6 field in the transcript.

\texttt{\textbackslash si\{\textit{unit}\}}\textsuperscript{siunitx}\textsuperscript{*}
Displays the unit with intelligent formatting.

\texttt{\textbackslash sigma}\textsuperscript{kernel command}\textsuperscript{* (maths mode)}
Greek letter sigma $\sigma$.

\texttt{\textbackslash sortart\{\textit{article}\}\{\textit{text}\}}
Example command.

\texttt{\textbackslash sortmediacreator\{\textit{first name(s)}\}\{\textit{surname}\}}
Example command.

\texttt{\textbackslash sortname\{\textit{first name(s)}\}\{\textit{surname}\}}
Example command.

\texttt{\textbackslash sortop\{\textit{text1}\}\{\textit{text2}\}}
Example command.

\texttt{\textbackslash sortvonname\{\textit{first name(s)}\}\{\textit{von}\}\{\textit{surname}\}}
Example command.

\texttt{\textbackslash space}\textsuperscript{kernel command}\textsuperscript{*}
Produces a space.

\texttt{\textbackslash SS}\textsuperscript{kernel command}\textsuperscript{*}
Produces the upper case eszett $\ss$.

\texttt{\textbackslash ss}\textsuperscript{kernel command}\textsuperscript{*}
Produces the lower case eszett $\ss$.

\texttt{\textbackslash string\{\textit{token}\}}\textsuperscript{\TeX primitive}\textsuperscript{*}
If \textit{token} is a control sequence it expands to the escape character followed by the control sequence name.

\texttt{\textbackslash strong\{\textit{text}\}}
Example command.

\texttt{\textbackslash subglossentry\{\textit{level}\}\{\textit{label}\}\{\textit{location list}\}}\textsuperscript{glossaries v3.08a+}
Used in the glossary to display a sub-entry.

\texttt{\textbackslash SUFFIXOF}\textsuperscript{bib2gls quark}
A quark to denote “is a suffix of” conditional in assign-fields conditionals. This token needs to be protected from expansion in the argument of \texttt{\textbackslash GlslXtrLoadResources}.
General Command Summary

\textbf{\verb|\surd|} \hfill kernel command\(^*\) (maths mode)
Surd symbol \(\sqrt{\cdot}\).

\textbf{\verb|\symbol{|\langle number\rangle}|} \hfill kernel command\(^*\)
Accesses the character identified by \langle number\rangle (use \textbf{\verb|\symbol{|\langle hex\rangle}|} if the number is hexadecimal).

\textbf{\verb|\tableofcontents|} \hfill kernel command\(^*\)
Displays the table of contents (by reading in the .toc file) and then opens .toc file to allow the sectioning commands to write to it.

\textbf{\verb|\tabularnewline[\langle len\rangle]|} \hfill kernel command
Tabular version of \verb|\\| (avoids conflict with forced line breaks in paragraph column formats).

\textbf{\verb|\texorpdffstring{|\langle \LaTeX \text{ code}\rangle}|{|\langle PDF \text{ text}\rangle}|} \hfill hyperref
Does \langle PDF \text{ text}\rangle if used in a PDF bookmark, otherwise does \langle \LaTeX \text{ code}\rangle.

\textbf{\verb|\textbf{|\langle text\rangle}|} \hfill kernel command
Displays the given text in bold.

\textbf{\verb|\textcolor{|\langle \text{ model}\rangle}|{|\langle \text{ spec}\rangle}|{|\langle \text{ text}\rangle}|} \hfill color
Typesets \langle text\rangle in the given colour.

\textbf{\verb|\text{|\langle \text{ language}\rangle}|{|\langle \text{ options}\rangle}|{|\langle \text{ text}\rangle}|} \hfill polyglossia
Typeset \langle text\rangle according to \langle language\rangle.

\textbf{\verb|\textsc{|\langle \text{ text}\rangle}|} \hfill kernel command
Applies small-caps font to \langle text\rangle.

\textbf{\verb|\textsf{|\langle \text{ text}\rangle}|} \hfill kernel command
Displays the given text in sans-serif.

\textbf{\verb|\textsmaller{|\langle \text{ text}\rangle}|} \hfill relsize
Typesets \langle text\rangle in a font size that's smaller than the surrounding text.

\textbf{\verb|\textstyle|} \hfill kernel command (maths mode)
Switch to in-line maths style (vertically compact).

\textbf{\verb|\textsubscript{|\langle text\rangle}|} \hfill kernel command\(^*\) as from 2015/01/01
Displays \langle text\rangle as a subscript.

\textbf{\verb|\textsuperscript{|\langle text\rangle}|} \hfill kernel command\(^*\)
Displays \langle text\rangle as a superscript.

\textbf{\verb|\texttt{|\langle \text{ text}\rangle}|} \hfill kernel command
Displays the given text in monospaced font.

\textbf{\verb|\textweathersymbol{|\langle number\rangle}|} \hfill ifsym
Displays weather symbol identified by \langle number\rangle.
General Command Summary

\TH \quad \text{kernel command}\text{*}
\begin{itemize}
\item Produces the upper case thorn Þ.
\end{itemize}

\th \quad \text{kernel command}\text{*}
\begin{itemize}
\item Produces the lower case thorn þ.
\end{itemize}

\texttt{\the\{register\}} \quad \text{\TeX primitive}\text{*}
\begin{itemize}
\item Expands \texttt{\{}\texttt{register}\texttt{\}} to the current value of the register.
\end{itemize}

\texttt{\theglossaryentry} \quad \text{glossaries}
\begin{itemize}
\item Textual representation of the glossaryentry counter, which is defined with the \texttt{entrycounter} option.
\end{itemize}

\texttt{\theHentrycounter} \quad \text{glossaries}
\begin{itemize}
\item When indexing, this is set to the \texttt{\the\{counter\}} command corresponding to the current indexing counter (or, if undefined, \texttt{\the\{counter\}}).
\end{itemize}

\texttt{\theHglossaryentry} \quad \text{glossaries}
\begin{itemize}
\item Hypertarget associated with the glossaryentry counter, which is defined with the \texttt{entrycounter} option.
\end{itemize}

\texttt{\TITLE\{(element-list)\}} \quad \text{bib2gls quark}
\begin{itemize}
\item A quark to denote a title case change in \texttt{assign-fields} syntax. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}.
\end{itemize}

\texttt{\toprule} \quad \text{booktabs}
\begin{itemize}
\item Horizontal rule for the top of a tabular-like environment.
\end{itemize}

\texttt{\TrackedLanguageFromDialect\{(dialect)\}} \quad \text{tracklang}
\begin{itemize}
\item Expands to the root language associated with the given (tracklang) dialect label.
\end{itemize}

\texttt{\TrackLangLastTrackedDialect} \quad \text{tracklang}
\begin{itemize}
\item Set by commands like \texttt{\TrackLocale}.
\end{itemize}

\texttt{\TrackLocale\{(language tag)\}} \quad \text{tracklang v1.3+}
\begin{itemize}
\item Tracks the given language tag.
\end{itemize}

\texttt{\transposefmt\{(maths)\}} \quad \text{bib2gls quark}
\begin{itemize}
\item Example command.
\end{itemize}

\texttt{\TRIM\{(element-list)\}} \quad \text{bib2gls quark}
\begin{itemize}
\item A quark to denote a trimmed element in \texttt{assign-fields}. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}.
\end{itemize}

\u \quad \text{kernel command}\text{*}
\begin{itemize}
\item Puts a breve accent over \texttt{\{}\texttt{character}\texttt{\}}.
\end{itemize}

\texttt{\u\{\texttt{hex}\}} \quad \text{bib2gls quark}
\begin{itemize}
\item A quark that identifies a character by its hexadecimal code in the values of some (but not all) resource options.
\end{itemize}

639
General Command Summary

\UC\{element-list\} \quad \text{bib2gls quark}

A quark to denote an upper case change in \texttt{assign-fields} syntax. This token needs to be protected from expansion in the argument of \texttt{\GlsXtrLoadResources}.

\undef\{cs\} \quad \text{etoolbox*}

Undefines the control sequence \langle cs\rangle.

\underline\{text\} \quad \text{kernel command}

Underlines the given text.

\unexpanded\{general text\} \quad \varepsilon\TeX\ primitive*

Expands to the argument.

\usepackage[\langle options\rangle][\langle name\rangle][\langle min version\rangle] \quad \text{kernel command*}

Loads the package identified by \langle name\rangle.

V

\vec\{character\} \quad \text{kernel command* (maths mode)}

Puts right arrow accent over \langle character\rangle.

\vecfmt\{symbol\} \quad \text{Example command.}

\vert \quad \text{kernel command* (maths mode)}

Vertical bar delimiter |.

W

\write18\{system call\} \quad \text{kernel command}

Perform shell escape if permitted.

X

\xglsaccsupp\{accessible text\}\{text\} \quad \text{glossaries-accsupp}

Used by the accessibility support to interface with the accsupp package, where \langle text\rangle is fully expanded.

\xGlsXtrSetField\{entry label\}\{field label\}\{value\} \quad \text{glossaries-extra v1.12+}

Globally assigns the (protected) full expansion of the given \langle value\rangle to the field identified by \langle field label\rangle for the entry identified by \langle entry label\rangle.

\xifinlist\{element\}\{list cs\}\{true\}\{false\} \quad \text{etoolbox}

Tests if the expansion of \langle element\rangle is in the list stored in the control sequence \langle list cs\rangle.

\xmakefirstuc\{text\} \quad \text{mfirstuc* v1.01+}

Applies \texttt{\makefirstuc} with one level expansion of the first token of \langle text\rangle.
Bibliography


Index

Symbols

! (boolean NOT) ................................................. 119
" (delimiter) .............................................. 50, 60, 115–122, 131, 177
" (hexadecimal identifier) ............................ 202, 278, 342
" (literal) .................................................. 16, 49, 115, 370
# (.bib string concatenation) ...................... 59, 114, 435
# (literal) ............................................. 16, 292, 293, 369
# (parameter) ........................................... 277, 288, 355
$ (maths shift) ........................................ 21, 46, 61, 157, 203, 209, 423
$ (regular expression) ................................ 16
% (comment) ........................................ 21, 124, 277, 487
% (literal) ........................................... 199, 292, 293
& (alignment) .......................................... 277
& (boolean AND) ....................................... 119, 148
& (literal) ........................................... 16, 292, 293, 370
' (apostrophe) ........................................... see apostrophe (')
( (start range) ........................................ 228, 229, 336
{ (begin group) ...................................... 21, 61, 101, 157, 178, 203, 277, 492
} (end group) ...................................... 21, 61, 101, 157, 178, 203, 277, 492
} (literal) ........................................... 292, 293
{ (literal) ........................................... 292, 293
) (end range) ........................................ 228, 229, 336
) (literal close parenthesis) .................... 16, 119
* (literal) ............................................. 16, 568
* (regular expression, zero or more) ........ 136
+ (literal) ........................................... 16, 568
+ (resource option concatenation) .......... 52, 114, 117–119, 148, 177
, (separator) ........................................ 52, 118
→ (field ref) ........................................... 119, 148, 178
− (literal) ............................................ 16, 568
. (end of sentence) .................................. see full stop (.)
. (regular expression, match any) ........... 136, 206
/ ..................................................... 16, 121
: (literal) ............................................ 16, 568
<= (comparison) .................................... 121–123
<> (comparison) ..................................... 120–122
< (comparison) ..................................... 118, 121–123, 148
< (literal) ........................................... 16, 568
< (regular expression) ......................... 116
= (assignment) .................................... 52, 118, 177
= (comparison) .................................... 120–122
>= (comparison) .................................. 121–123
> (comparison) ..................................... 118, 121–123
> (literal) ........................................... 16, 568
> (regular expression) ......................... 116
? (literal) ........................................... 16, 568
? (regular expression) ......................... 116
? (regular expression, zero or one) ........ 116
@ (bib entry identifier) ......................... 50, 184
[ (literal open square bracket) ............... 16, 569
[ (optional) ....................................... 118, 119
\{ ................................................. 199, 277, 292, 293, 567
\} .................................................. 199, 277, 292, 293, 567
" (quark) ........................................... 16, 568
" .................................................. 27, 115, 157, 567
# .................................................. 199, 277, 288, 292, 293, 567
# (regexp (or regex) quark) ................. 16
\$ .................................................. 199, 567
\$ (regexp (or regex) quark) ................. 16
\% .................................................. 199, 200, 277, 292, 293, 567
\& .................................................. 199, 277, 292, 293, 567
\& (regexp (or regex) quark) ................. 16
\' .................................................. 288, 567
\ (regexp (or regex) quark) ................. 16

see command line options (bib2gls)
& command line options (convertgls2bib)

643
Index

\) (regexp (or regex) quark) .... 16 - (non-breakable space) .... 21, 35, 36, 47, 157, 203–206, 215, 335, 398, 539
\* (regexp (or regex) quark) .... 16
\+ (regexp (or regex) quark) .... 16
\, .................................................. 485, 568
\- (regexp (or regex) quark) .... 16
\. ............................................. 136, 159
\. (regexp (or regex) quark) .... 16
\/ (regexp (or regex) quark) .... 16
\: (regexp (or regex) quark) .... 16
\< (regexp (or regex) quark) .... 16
\> (regexp (or regex) quark) .... 16
\? (regexp (or regex) quark) .... 16
\@ .......................................... 248, 338, 568
\@currentHref ..................... 40, 568
\@currentlabelname ............. 40, 568
\@firstofone ................. 350, 568
\@for .................................. 367, 568
\@gls@hypergroup .............. 32
\@gobble .......................... 350, 569
\@istfilename .................. 19, 569
\[ (regexp (or regex) quark) .... 16
\[ ....................................... 199, 569
\[ (regexp (or regex) quark) .... 16
\] (regexp (or regex) quark) .... 16
\^[ ] ................................ 68, 569
\^[ ] (regexp (or regex) quark) .... 16
\^- ...................................... 199, 277, 292, 293, 569
\]- (regexp (or regex) quark) .... 16
\]- (regexp (or regex) quark) .... 16
\[ (regexp (or regex) quark) .... 16
\[ (regexp (or regex) quark) .... 16
\[ (escape) ..................... 21, 61, 157, 203, 284–286, 492
\[ (literal) .................... 16, 101, 277
\] (literal close square bracket) .... 16, 569
\] (optional) ...................... 118
\^[ ] (literal) .................. 16, 370
\^[ ] (regular expression) .... 16
\^[ ] (superscript) .............. 21, 38
\^[ ] (literal) .................. 292, 293, 369
\^[ ] (subscript) .............. 21, 38, 277
\] (boolean OR) ............... 119
\] (regular expression or) .... 15
\- (literal) ...................... 16, 277

A

abbreviation styles
long-em-short-em ..................... 70
long-hyphen-postshort-hyphen .... 614
long-noshort ...................... 615
long-noshort-desc ............... 69
long-only-short-only ............ 463, 465, 544
long-only-short-only-desc ....... 616
long-postshort-user-desc ....... 95
long-short ...................... 212
long-short-desc ............ 93, 94, 512, 526, 543
long-short-sc .............. 69, 212, 226
long-short-sm .............. 69
long-short-user ............. 93, 520
long-short-user-desc .......... 95
short-hyphen-long-hyphen ....... 619
short-long ..................... 173
short-long-desc .......... 619
short-long-user-desc .......... 620
short-nolong .................. 620
short-nolong-desc ............. 619
\abbreviationname ............ 570
\abbrvpluralsuffix ............ 226, 570
see also \glxtababbrvpluralsuffix
\ac ............................................... 47, 570
\acronymname .................. 570
\acronymtype .................. 322, 327, 373, 570
\acrpluralsuffix .......... 226, 570
\AE ........................................ 209, 213, 570
\ae ........................................ 209, 213, 570
\alph .................................. 243, 570
\Alpha .................................. 570
\alpha .................................. 278, 570
animals.bib .................. 442, 449, 452, 524
\alsoname ..................... 376, 377, 570
applications ............... 49, 226, 370
arara .................................. 19

644
Index
bibtex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
convertgls2bib . . . . . . . . . . 372, 375–382
kpsewhich . . . . . . . . . . . 17, 20, 34, 132, 372
makeglossaries . . . . . . . . . . . . . . . . . 1, 19
makeglossaries-lite . . . . . . . . . . . . . 19
makeindex . . . . . . . . . . . . . . 17, 35, 261, 285
xindy . . . . . . . . . . . . . . . . . . . . 4, 35, 261, 285
\approx . . . . . . . . . . . . . . . . . . . . . . . . . 276, 570
\appto . . . . . . . . . . . . . . . . . . . . . . . . . . 287, 570
\apptoglossarypreamble . . . . 248, 287, 570
see also \glossarypreamble

ascii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 288
\AtEndDocument . . . . . . . . . . . . . . . . . . . 1, 570
attributes
see category attributes

\autoref . . . . . . . . . . 209, 571, see also \label
B
\backmatter . . . . . . . . . . . . . . . . . . . . . 229, 571
bacteria.bib . . . . . . . . . . . . 107, 393, 463, 524
baseunits.bib . . . . . . 395, 397, 467, 469, 473,
524, 527

\bfseries . . . . . . . . . . . . . . . . . . . . . . . 475, 571
bib2gls-en.xml . . . . . . 18, 186, 335, 338, 344
bib2gls.bat . . . . . . . . . . . . . . . . . . . . . . . . . 20
bib2gls.jar . . . . . . . . . . . . . . . . . . . . . . . 20, 28
bib2gls.sh . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
\bibglsaliassep . . . . . . . . . . . . . . . . . . . . 334
\bibglsampersandchar . . . . . . . . . . . 23, 370
\bibglsaposchar . . . . . . . . . . . . . . . . . . . . 370
\bibglscircumchar . . . . . . . . . . . . . . . 23, 370
\bibglscompact . . . . . . . . . . . . . . . . . 244, 335
\bibglscontributor . . . . . . 23, 197, 198, 368
\bibglscontributorlist . . . . . 23, 197, 367
\bibglscopytoglossary . . . . . . . . . 130, 364
\bibglsdate . . . . . . . . . . . . . . . . . . 23, 203, 368
\bibglsdategroup . . . . . . . . . . . 351, 359, 485
\bibglsdategrouphier . . . . . . . . . . . . . . 359
\bibglsdategrouptitle . . . . . 351, 359, 485
\bibglsdategrouptitlehier . . . . . . . . 359
\bibglsdatetime . . . . . . . . . . . . . 23, 203, 368
\bibglsdatetimegroup . . . . . . . . . . 350, 358
\bibglsdatetimegrouphier . . . . . 358, 572

\bibglsdatetimegrouphierfinalargs
358

\bibglsdatetimegrouptitle . . . . 350, 358
\bibglsdatetimegrouptitlehier 358, 572
\bibglsdatetimegrouptitlehierfinalargs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 358
\bibglsdatetimeremainder . . . . . . . . . 368
\bibglsdefcompoundset . . . . . 107, 318, 333
\bibglsdefinitionindex . . . . . . . . 165, 371
\bibglsdelimN . . . . . . . . . . . . . . . . . . . . . . 334
see also \bibglslastDelimN

\bibglsdollarchar . . . . . . . . . . . 23, 200, 370
\bibglsdoublequotechar . . . . . . . . . . . . 370
\bibglsdualprefixlabel . . . . . . . . . . . . 369
\bibglsemptygroup . . . . . . . . . . . . . . 349, 356
\bibglsemptygrouphier . . . . . . . . . . . . . 356
\bibglsemptygrouptitle . . . . . . . . 349, 357
\bibglsemptygrouptitlehier . . . . . . . 357
\bibglsexternalprefixlabel . . . . . . . 369
\bibglsfirstuc . . . . . . . . . . . . . . 23, 207, 371
\bibglsflattenedchildpostsort . . . 363
\bibglsflattenedchildpresort . 147, 363
\bibglsflattenedhomograph . . . . 147, 362
\bibglsgrouplevel . . . . . . . . . . . . . . . . . 355
\bibglshashchar . . . . . . . . . . . . . . . . . 23, 369
\bibglshexunicodechar . . . . . . . . . 202, 342
\bibglshiersubgrouptitle . . 151, 355, 362
\bibglshrefchar . . . . . . . . . . . . . . . . 256, 342
\bibglshrefunicode . . . . . . . . . . . . 256, 342
\bibglshypergroup . . . . . . . . . . . . . . . . . 345
\bibglshyperlink . . . . . . . . . . . . 23, 312, 365
\bibglsinterloper . . . . . . . . . . . . . . 240, 336
\bibglslastDelimN . . . . . . . . . . . . . . . . . 335
see also \bibglsdelimN

\bibglslettergroup . . . . . . . . . 345, 346, 355
\bibglslettergrouphier . . . . . . . . . . . . 355
\bibglslettergrouptitle . . . 346, 347, 356
\bibglslettergrouptitlehier . . . . . . 356
\bibglslocationgroup . . . . . . . . . . 338, 340
\bibglslocationgroupsep . . . . . . . 339, 340
\bibglslocprefix . . . . . . . 246, 248, 337, 338
\bibglslocsuffix . . . . . . . . . . . . . . . . . . . 338
\bibglslowercase . . . . . . . . . . . . 23, 206, 370
\bibglsmergedgroup . . . . . . . . . . . . 354, 361

645


Index

\bibglssetemptygrouptitlehier \ 356
\bibglssetlastgrouptitle \ 344, 345
\bibglssetlettergrouptitle \ 151, 345, 346, 355
\bibglssetlettergrouptitlehier \ 151, 355
\bibglssetlocationrecordcount \ 365
\bibglssetmergedgrouptitle \ 354, 360
\bibglssetmergedgrouptitlehier \ 360
\bibglssetnumbergrouptitle \ 349, 357
\bibglssetnumbergrouptitlehier \ 357
\bibglssetothergrouptitle \ 348, 356
\bibglssetothergrouptitlehier \ 358
\bibglssetrecordcount \ 364
\bibglssettimegrouptitle \ 351, 359
\bibglssettimegrouptitlehier \ 359
\bibglssettotalrecordcount \ 364
\bibglssetunicodegrouptitle \ 344, 352–354, 360
\bibglssetunicodegrouptitlehier \ 360
\bibglssetwidest \ 125, 365–367
\bibglssetwidelast fallback \ 125, 366
\bibglssetwidestforgtype \ 126, 366, 367
\bibglssetwidestforgtype fallback \ 125, 366
\bibglssetwidesttoplevel fallback \ 125, 367
\bibglssetwidesttoplevel forgtype fallback \ 125, 367
\bibglssetsupplemental \ 256, 341
\bibglssetsupplementalsep \ 341
\bibglssetsublist \ 256, 341
\bibglssetsubsept \ 257, 342
\bibglsettertiaryprefixlabel \ 369
\bibglsetime \ 23, 203, 368
\bibglsetimegroup \ 352, 359
\bibglsetimegrouphier \ 359
\bibglsetimegroup \ 352, 360
\bibglsetimegrouphier \ 360
\bibglsetitlecase \ 23, 207, 371
\bibglsetunderscorechar \ 23, 369
\bibglsetunicodegroup \ 295, 344, 352, 354, 360
\bibglsetunicodegrouptitle \ 360
\bibglsetunicodegrouphier \ 360
\bibglsetunicodedegroup \ 344, 352–354, 360
\bibglsetunicodedegrouptitle \ 344, 352–354, 360
\bibglsetunicodedegrouptitlehier \ 360
\bibglsetuppercase \ 23, 206, 370
\bibglsupcase \ 83, 324
\bibglsealias \ 245, 334
\bibglseindex \ 166, 371
\bibglsselongfont \ 83, 324, 328
\bibglsetimesee \ 245, 334
\bibglsetimeseealso \ 245, 334
\bibliography \ 100, 581
\bigmathsymbols.bib \ 419, 499, 501
\bigoperate
ontnam
efmt \ 419
\binaryoperators.bib \ 425, 499
\boldsymbol \ 25, 582
\books.bib \ 404, 407, 478, 479, 490
\boolean values see conditionals and boolean values
\bottomrule \ 475, 582

C
\c \ 67, 582
\capitalisewords \ 206, 209, 210, 371, 582
\caption \ 41, 582
\case-change \ 206–210, 215
see also upper case, lower case, title case
& sentence case
\CAT (quark) \ 16, 120
\category attributes \ 463
\aposplural \ 226
\externallocation \ 254–257
\glossdesc \ 206, 515
\glossdescfont \ 465
\glossname \ 24, 206, 435, 466, 514
\glossnamefont \ 465, 466, 492, 520, 521, 526, 530
\glosssymbolfont \ 530
\markwords \ 621
\nohyperfirst \ 562
\noshortplural \ 226
\recordcount \ 42
\targetname \ 152, 155
\targeturl \ 152, 155
\textformat \ 465, 492, 520, 526

647
Index

\ce ................................................. 389, 582
\chapter ........................................ 582
\chapter* ........................................ 236–240
\chapter* ........................................ 472, 582
\char ............................................. 21, 22, 278, 377, 582
chemicalformula.bib 389, 460, 524, 527, 528
\citation ......................................... 38, 98, 413, 494, 582
citations.bib .................................... 413, 494
\cite ............................................... 100, 494, 496, 582
\CJK ............................................... 347
\CJK environment ................................ 347
\cjkname ......................................... 347
\cldr (Unicode Common Locale Data Repository) .................................. 28, 273, 399, 485
\color ............................................ 112, 582
command line options (bib2gls)
  --break-space ................................ 35, 203, 398, 539
  --cite-as-record ................................ 38, 98, 413, 494
  --collapse-same-location-range 38, 228
  --custom-packages 22, 36, 37, 207, 260, 279, 284
  --debug ...................................... 17, 22, 29, 193, 231, 281
  --dir .......................................... 34, 132
  --expand-fields 45, 342
  --force-cross-resource-refs 14, 48
  --help ......................................... 29
  --ignore-packages ................................ 36
  --interpret .................................... 21, 36, 101
  --list-known-packages 22, 36
  --locale ....................................... 18, 30, 260
  --log-file ..................................... 17, 35
  --map-format .................................. 39, 231, 250
  --merge-nameref-on 4, 40, 41, 228, 230, 256
  --merge-wrglossary-records 41, 251
  --mfirstuc-math-protection 13, 24, 46
  --mfirstuc-protection 13, 24, 45, 46
  --nested-link-check 13, 46, 47
  --no-break-space 36
  --no-cite-as-record 38
  --no-collapse-same-location-range 39
  --no-debug .................................... 22, 29
  --no-expand-fields 45
  --no-force-cross-resource-refs 48
  --no-group .................................... 34, 151, 169, 295
  --no-interpret ................................ 14, 21, 36
  --no-merge-wrglossary-records 41
  --no-mfirstuc-math-protection 46
  --no-mfirstuc-protection 45, 46
  --no-nested-link-check 47
  --no-obey-aux-catcode 38
  --no-provide-glossaries 11, 49, 155, 172, 296
  --no-record-count 43
  --no-record-count-unit 43
  --no-replace-quotes 49
  --no-retain-formats 44
  --no-support-unicode-script 38
  --no-trim-fields 48
  --no-verbose 29
  --no-warn-non-bib-fields 44
  --no-warn-unknown-entry-types 45
  --obey-aux-catcode 38
  --packages ................................... 36, 37, 207
  --provide-glossaries 11, 48, 452
  --quiet ........................................ 30
  --record-count 42, 43, 262–265, 364
  --record-count-rule 42, 43
  --record-count-unit 43, 365
  --replace-quotes 49, 370
  --retain-formats 44
  --shortcuts 47
  --silent ....................................... 30
  --support-unicode-script 37
  --tex-encoding 35, 124
  --trim-except-fields 48
  --trim-fields 47, 48
  --trim-only-fields 48
  --verbose ..................................... 26–29, 193, 221, 274, 283
  --version ...................................... 29
  --warn-non-bib-fields 8, 44, 52, 500
  --warn-unknown-entry-types 45

648
Index

command line options (convertgls2bib)

--absorb-see .................................. 375
date-time group .............................. 31, 350, 358
date group .................................. 31, 351
\DeclareOptions ................................ 36, 583
\DeclareOptions* ............................... 36, 583
def ........................................ 367, 583
delimN .......................................... 230, 334, 583
delimR ......................................... 241, 244, 245, 583
derivedunits.bib ................................. 397, 467, 469, 473, 524, 527
description environment ..................... 526
descriptionname ................................ 583
detokenize .................................... 132, 583
dGls ........................................... 82, 87, 583

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix

dgls ........................................... 82, 87, 583

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
dglsdisp ....................................... 210, 583, see also dglsdisp
dglslink ........................................ 210, 583, see also dglsdisp
dGlsp ........................................... 87, 584

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
dGlsp ........................................... 87, 584

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix

\DH ............................................. 209, 584
\dh ............................................. 209, 584
diamondsuit .................................... 7, 584
digraph ......................................... 32

\displaystyle .................................. 419, 584
\DJ ............................................. 209, 584
\dj ............................................. 209, 584
document environment ......................... 375

DTLandname ..................................... 480, 584

see also \DTLformatlist

\DTLformatlist ................................. 197, 367, 480, 584

see also \ifDTListsempty

\DTLlistformatlistsep .......................... 584

see also \DTLformatlist

\DTLlistformatoxford .......................... 480, 584

see also \DTLformatlist

\DTMdisplaydate ............................... 399, 539, 584
dual ............................................ 71

date-time group .............................. 31, 350, 358
date group .................................. 31, 351
\DeclareOptions ................................ 36, 583
\DeclareOptions* ............................... 36, 583
def ........................................... 367, 583
delimN .......................................... 230, 334, 583
delimR ......................................... 241, 244, 245, 583
derivedunits.bib ................................. 397, 467, 469, 473, 524, 527
description environment ..................... 526
descriptionname ................................ 583
detokenize .................................... 132, 583
dGls ........................................... 82, 87, 583

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
dgls ........................................... 82, 87, 583

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
dglsdisp ....................................... 210, 583, see also dglsdisp
dglslink ........................................ 210, 583, see also dglsdisp
dGlsp ........................................... 87, 584

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
dGlsp ........................................... 87, 584

see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix

\DH ............................................. 209, 584
\dh ............................................. 209, 584
diamondsuit .................................... 7, 584
digraph ......................................... 32

\displaystyle .................................. 419, 584
\DJ ............................................. 209, 584
\dj ............................................. 209, 584
document environment ......................... 375

DTLandname ..................................... 480, 584

see also \DTLformatlist

\DTLformatlist ................................. 197, 367, 480, 584

see also \ifDTListsempty

\DTLlistformatlistsep .......................... 584

see also \DTLformatlist

\DTLlistformatoxford .......................... 480, 584

see also \DTLformatlist

\DTMdisplaydate ............................... 399, 539, 584
dual ............................................ 71
Index

E
\edef ........................................... 519, 584
\eglsupdatewidest ................................ 584
  see also \glssetwidest & \eglsupdatewidest
\em .............................................. 479, 584
\emph ........................................... 39, 40, 585
empty group (unknown commands) ............. 31, 349
ENCAP (encapsulating command) ............... 108, 227, 228, 232, 256
encoding ...................................... 35, 50, 124, 372
\endfoot ....................................... 475, 585
\endhead ....................................... 475, 585
\ensuremath .................................. 209, 423, 426, 585
entry types
  @abbreviation ................................ 69, 70, 104, 215–219, 223, 224, 275, 276, 321, 393, 435, 437
  @acronym .................................... 70, 104, 215–219, 223, 224, 275, 322, 380
  @bibtxentry 11, 38, 55, 70, 71, 97, 98, 222, 275, 276, 328, 404, 413, 496, 497
  @comment .................................... 51
  @compoundset ................................ 57, 58, 107, 109, 315–318
  @contributor ................................. 55, 70, 71, 98, 275, 276, 328, 496
  @dualabbreviation .. 54, 90, 91, 95, 310–313, 327
  @dualabbreviationentry 69, 73, 75, 78, 87, 88, 311–314, 325
  @dualacronym ................................ 95, 327
  @dualentry ................................. 50, 78, 87–90, 170, 220, 303–305, 309–314, 322, 544
  @dualentryabbreviation .......... 87, 88, 314, 325, 326
  @dualabbreviationentry 72–75, 82, 96, 312, 314, 324, 527
  @dualindexentry 16, 72, 80–83, 97, 105, 311, 314, 322
  @dualindexabbreviation 87, 311, 314, 323
  @dualindexnumber .......................... 87, 311, 314, 323
  @dualindexsymbol ......................... 72, 83, 86, 87, 311, 314, 323
  @dualnumber .............................. 89, 326
  @dualsymbol .............................. 71, 89, 311, 313, 326, 473
@entry ........................................ 25, 65–67, 78–80, 104, 170, 208, 217–221, 275, 276, 319, 375, 390, 398, 435, 442–446, 462, 469, 482, 484, 544
@index ........................................... 16, 67–72, 80, 98, 104, 105, 144, 149, 170, 179, 193, 195, 212, 215–221, 259, 275, 276, 281, 317, 318, 321, 375, 399, 437, 446–449, 482, 509, 527, 528, 544
@indexplural .................................. 68, 104, 193, 195, 218, 220, 275, 281, 321, 509
@number ....................................... 66, 105, 221, 223, 320, 455
@preamble .................................... 11, 14, 22, 25, 26, 60, 61, 111–114, 124, 147, 192, 207, 259, 284, 363, 385, 387, 413, 415, 435, 437, 514
@progenitor 11, 54, 56, 101–105, 172, 329
@spawnabbreviation ................. 104, 330, 331
@spawncronym ................................ 104, 331
@spawndualindexentry ............. 97, 104, 105, 332, 333
@spawntentry ................................ 104, 330
@spawnpindex ................................ 104, 329
@spawnpindexplural .................. 104, 329, 330
@spawnnnumber ......................... 105, 332
@spawnsymbol .............................. 105, 331, 332
@string ......................................... 12, 59, 435, 437
@symbol .................................... 24, 25, 66, 71, 105, 221, 223, 259, 276, 281, 320, 390, 395, 414, 419, 426, 432, 462, 469, 509, 544
@tertiaryindexabbreviationentry 78, 96, 312, 314, 327, 328
equation counter ......................... 250, 253

F
fields
  access ......................................... 54, 592
  adoptparents ............................. 54, 101, 102, 105, 172
Index

descriptionaccess 55, 592
descriptionplural 53–57, 78, 310
descriptionpluralaccess 55, 593
dualdescription 55, 90–95, 215, 305, 311, 327
duallong 54, 90, 91
dualprefix 54, 71, 78–84, 87–91, 204, 205
dualprefixfirst 54, 71, 78–84, 87–91, 204, 205
dualprefixfirstplural 54, 71, 78–84, 87–91, 204, 205
dualprefixplural 54, 71, 78–84, 87–91, 204, 205
dualshort 54, 90, 91, 215, 305, 311, 327
dualshortplural 54, 90, 91, 227
elements 57, 58, 109
first 47, 53–55, 174, 175, 179, 187, 217, 218, 226, 399, 484, 492, 538, 545
firstaccess 55, 593
firstplural 47, 53–57, 175, 218, 225, 226
firstpluralaccess 55, 593
longaccess 55, 593
longplural 47, 53–57, 87, 90, 91, 225, 226
longpluralaccess 55, 593
main 58, 109
nonumberlist 13, 53, 57, 232
option 58, 109
plural 47, 53–55, 68, 78, 83, 87, 89, 104, 175, 217, 218, 225, 226, 275, 310
pluralaccess 55, 594
prefix 54, 78–84, 87–91, 204, 205
prefixfirst 54, 78–84, 87–91, 204, 205
prefixfirstplural 54, 78–84, 87–91, 204, 205
prefixplural 54, 78–84, 87–91, 204, 205
plural access 55, 594
see 13, 14, 21, 52, 53, 133, 134, 141, 147, 154, 156, 163, 166, 167, 182, 183, 190, 193–196, 227, 231, 245, 246, 259, 310, 333, 334, 372, 376, 377
seealso 14, 21, 52, 53, 133, 141, 147, 154, 156, 163, 167, 183, 190–196, 227, 228, 231, 245, 259, 334, 372, 376, 377
shortaccess 55, 594
shortplural 47, 53–57, 87, 90, 225–227
shortpluralaccess 55, 595
symbol 45, 47, 53, 55, 83, 84, 89, 177, 274, 311, 390, 395, 469, 475, 527–531, 543
symbolaccess 55, 595
symbolplural 53, 55, 83, 89
symbolpluralaccess 55, 595
textaccess 55, 595
user1 53, 57, 61, 62, 67, 90, 93, 116, 122, 158, 167, 174, 192, 223, 293, 457, 458, 479, 482, 485, 503, 527
user2 53, 57, 174, 458, 479–482
user3 53, 57, 174, 458, 482, 484
user4 53, 57
user5 53, 57
user6 53, 57
Index

fields, internal
  bib2gls@sort ........................................ 52, 57
  bib2gls@sortfallback ......................... 57
  bibtextcontributor ....................... 55
  bibtextentry .......................... 55, 71, 98, 497
  bibtextentry@{entry-type} .............. 55, 71, 98
  bibtextype .................................. 55, 98
  childcount .................................. 55, 138
  childlist .................................... 55, 138, 140
  counter ...................................... 13, 55
  currcount .................................... 57
  definition@{value} ....................... 57
  desc ............................................ 57
  descplural .................................. 57
  dual .......................................... 55, 305, 313
  {field}endpunc ......................... 56, 186, 187, 492, 518
  firstpl ..................................... 57
  flag .......................................... 57
  group .......................................... 4, 5, 8, 9, 12, 21, 30, 31, 34, 44, 55, 56,
                                           96, 128, 129, 144, 148, 156, 169, 182, 259,
                                           294–296, 343–346, 353, 450, 467, 469, 472,
                                           496, 500, 503, 506, 543
  index ......................................... 57
  indexcounter ................................ 56, 250–253
  indexed ....................................... 43, 57, 250
  level ........................................... 57
  location ..................................... 56, 102, 227, 230–235, 239, 240,
                                           245, 500
  loclist ...................................... 56, 227–233
  longpl ....................................... 57
  originalentrytype ..................... 56, 126, 184, 185
  originalid .................................. 56, 165
  prenumberlist ............................. 57
  prevcount .................................. 57
  prevcount@{value} ....................... 57
  prevunitmax ................................ 57
  prevunittotal ............................. 57
  primarylocations ..................... 56, 232–234,
                                           237–240, 340
  progenitor .................................. 56, 102
  progeny ...................................... 56, 102
  recordcount ............................ 42, 43, 56, 264, 364
  recordcount.@{counter} .............. 42, 56, 364
  recordcount.@{counter}.@{location} .... 43,
                                           56, 365
  rootancestor ......................... 56, 140
  secondarygroup ......................... 56, 182, 296
  secondarysort ......................... 56, 296
  shortpl ..................................... 57
  siblingcount ......................... 56, 140
  siblinglist ......................... 56, 140
  sort .......................................... 17, 24–26, 44, 50–52, 55–57, 60, 61,
                                           66–69, 83, 88, 91, 97, 110, 116, 200,
                                           216–224, 259, 267, 273–276, 280, 283, 290,
                                           296, 319, 320, 344, 347, 373, 374, 390, 395,
                                           414, 462, 466, 475, 509, 528
  sortvalue ..................................... 57
  type .......................................... 4, 5, 9–13, 21, 32, 33, 44, 49, 55, 56, 79,
                                           96–98, 101, 125–129, 136, 149, 156,
                                           169–172, 186, 247, 303–305, 314, 317, 318,
                                           321, 322, 327, 344, 353, 365, 373, 374, 377,
                                           450–453, 494, 542
  unitlist ..................................... 57
  useindex ..................................... 56, 166, 200
  user ............................................ 57, 458, 519, 520
  userii ........................................ 57
  useriv ........................................ 57
  userv .......................................... 57
  uservi ........................................ 57

file formats
  .aux .......................................... 1, 18–21, 32–35, 42, 47, 110, 133, 154,
                                           227, 230, 231, 251, 257, 261, 264, 277
  .bat ............................................. 20
  .bbl ............................................ 413
  .bib .......................................... 1, 11, 19, 21, 34, 45, 50, 112, 113, 124,
                                           132, 154, 163, 306, 372
  .gig .......................................... 25, 34, 35
  .gls ............................................. 343
  .glstex ........................................ 11, 14, 19, 32, 34, 42, 45, 47, 52, 55,
                                           56, 61, 74, 91, 96, 110, 114, 124, 128, 132,
                                           133, 155, 156, 260, 305, 306, 319, 343, 413
  .jar ............................................ 20
  .log ............................................. 36, 345
  .out ............................................. 20
  .sh .............................................. 20
  .tex .......................................... 1, 372

652
Index

\glossaryheader
films.bib 385, 407, 478, 479
first use 597
first use flag 599
\FIRSTLC (quark) 16, 117
\FIRSTUC (quark) 16, 117
\footnote 585
\forallall 426, 585
\forallglossaries 585
\forallglsentries 585
\forallglossaries & \forallglossyentries
\forlistloop 520, 585
\frontmatter 229, 586

G

\glosstocbib
\glossariesextrasetup 586
glossary styles 6
altlist 457, 494, 526
altlistgroup 486
alttree 124, 125, 432, 457, 500, 508
alttreegroup 462, 506
bookindex 233, 465, 486, 487, 521, 530
index 30, 465
indexgroup 3, 30, 34, 494
indexhypergroup 32, 343, 345
list 452, 457, 475
long 452
long3col–booktabs 475
mcolalttree 499
mcolalttreegroup 462, 528
mcolindexgroup 469
super 563
topic 9, 511
tree 235, 563
treeroot 9, 450, 453
treenoname 605
glossaryentry counter 561\glossaryheader 475, 586
\glossaryname 586
\glossarypostamble 586
see also \apptoglossarypreamble
\glossarypreamble 287, 586
see also \apptoglossarypreamble
glossarysubentry counter 565
glossentry 230, 475, 586
\Glossentrydesc 201, 586
\glossentrydesc 206, 586
\glossentryname 586
\glossentryname 206, 466, 586
\glossentrynameother 466, 530, 531, 586
\Glossentrysymbol 586
\glossentrysymbol 528–531, 586
\glossxtrsetpopts 586
\GLS 210, 587
\Gls 45, 210, 587
\glslong 42, 47, 52, 72, 86, 105, 133, 147, 154, 161, 162, 210, 217, 225, 228, 244, 249, 251, 303, 336, 372, 484, 518, 587
\glsabbrvdefaultfont 587
\glsabbrvemfont 587
\glsabbrvfont 512, 587
\glsabbrvhyphenfont 587
\glsabbrvonlyfont 463, 587
\glsabbrvscfont 587
\glsabbrvuserfont 587
\glssaccessdesc 587
\glsaccessdesc 587
\glsaccessdesc 587
\glsaccessdesc 587
\glsaccessdesc 587
\glsaccessdesc 587
\glsaccessdesc 587
\glsaccessdesc 587
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessfirst 588
\glsaccessname 588
\glsaccessname 589
\glsaccessname 589
\glsaccessplural 589
Index

\glsaccessplural 589
\Glsaccessshort 589
\glsaccessshort 589
\Glsaccessshortpl 589
\glsaccessshortpl 589
\Glsaccesssymbol 589
\glsaccesssymbol 589
\Glsaccesssymbolplural 589
\glsaccesssymbolplural 589
\Glsaccesstext 590
\glsaccesstext 590
\glsaccsupp 590

see also \glsaccsupp
\glsadd 1, 85, 228, 230, 235–239, 253, 254, 336, 590
counter 236, 590
format 229, 236, 254, 336
theHvalue 254
thevalue 253, 254
\glsaddall 1, 133, 590
\glsaddallunused 229, 590
\glsaddeach 590
\glsaddkey 51, 137, 174, 590
\glsaddstoragekey 51, 98, 137, 516, 590
\glsautoprefixkey 591
\glsbackslash 23, 200, 277, 591
\glscapturedgroup 116, 159, 591
\glscategory 97, 591
\glsclosebrace 23, 591
\glscurrententrylabel 484, 591
\glscurrentfieldvalue 138, 591
\glsdefaulttype 373, 377, 591
\glsdefpostdesc 67, 459, 591
\glsdefpostlink 591
\glsdefpostname 479, 591
\glsdesc 546, 591
\glsdescriptionaccessdisplay 591
\glsdescriptionpluralaccessdisplay 591
\glsdescwidth 475, 591
\glsdisablehyper 592
\glsdisp 210, 592, see also \glslink
\glsdoifexists 592
\glsdoifexistssordo 592
\glsdoifnoexists 592
\glsdoifnoexistssordo 592
\glsenablehyper 592
\glsendrange 228, 592
\glsentryaccess 592
\glsentrycounterlabel 592
\GlsEntryCounterLabelPrefix 592
\glsentrydesc 206, 592
\glsentrydesc 592
\glsentrydescaccess 592
\glsentrydescplural 593
\glsentrydescplural 593
\glsentrydescpluralaccess 593
\glsentryfirst 23, 593
\glsentryfirst 23, 593
\glsentryfirstaccess 593
\glsentryfirstplural 23, 593
\glsentryfirstplural 23, 593
\glsentryfirstpluralaccess 593
\glsentryitem 475, 593
\glsentrylong 23, 593
\glsentrylong 23, 466, 593
\glsentrylongaccess 593
\glsentrylongpl 23, 593
\glsentrylongpl 23, 593
\glsentrylongpluralaccess 593
\glsentryname 23, 206, 593
\glsentryname 17, 23, 100, 168, 190, 192, 312, 497, 594
\glsentryplural 23, 594
\glsentryplural 23, 594
\glsentrypluralaccess 23, 594
\glsentrypluralaccess 23, 594
\glsentrypluralaccess 23, 594
\glsentryprefix 594
\glsentryprefix 594
\glsentryprefix 594
\glsentryprefix 594
\glsentryprefixfirst 594
\glsentryprefixfirst 594
\glsentryprefixfirst 594
\glsentryprefixfirstplural 594
\glsentryprefixfirstplural 594
\glsentryprefixfirstplural 594
\glsentryprefixplural 594
\glsentryprefixplural 594
\glsentryprefixplural 594
\glsentryshort 23, 594
\glsentryshort 23, 594
\glsentryshortaccess 594
\glsentryshortaccess 594
\glsentryshortpl 23, 594

654
Index
Index

thevalue .......................... 590
\wrgloss .......................... 600
\glslocalreset .................... 600
\glslocalunset ..................... 599, 600
\glslongaccessdisplay .......... 600
\glslongdefaultfont .......... 600
\glslongemfont .................... 600
\glslongextraSetWidest ........... 125, 600
see also \glslongextraUpdateWidest
\glslongextraUpdateWidest ........ 600
\glslongfont ....................... 512, 600
\glslongfootnotefont ............. 600
\glslonghyphenfont ............... 600
\glslongonlyfont .............. 463, 600
\glslongpluralaccessdisplay ..... 600
\glslongtok ....................... 512, 601
\glslonguserfont ............... 313, 601
\glslowercase .................... 370, 601
\glsname ......................... 601
\glsnameaccessdisplay ........... 601
\glsnamefont ..................... 252, 466, 601
\glsnavhypertarget ............. 33, 601
\glsnoexpandfields .......... 45, 377
\glsnoidxdisplayloc .......... 228, 229
\glsnoidxloclist .................. 230, 601
\glsnoidxloclisthandler ........ 230, 601
\glsnumberformat ................ 39, 40, 162, 230–232, 242, 243, 251, 257, 601
\glsnumbersgroupname ........... 350, 601
\glsopenbrace ................... 23, 601
\glspatchtabularx .............. 601
\glspercentchar .............. 23, 602
\GLSp1 ..................... 602
\Glsp1 .......................... 602
\glspl ..................... 217, 225, 602
\glsplural ....................... 217, 602
\glspluralaccessdisplay ...... 602
\glspluralsuffix .............. 217, 218, 225, 226, 602
\glspost–inline .............. 602
\glspostdescription ........... 476, 602
\glsps ....................... 46, 602
\glspt ..................... 602
\glsquote ....................... 602
\glsrefentry ................. 602, see also \ref
\glsrenewcommand ............ 319, 602
\glsreset ....................... 602
\glsresetentrycounter ......... 602
\glssee ....................... 52, 261, 375, 603
\glsseefirstitem ............ 52, 261, 375, 603
see also \glsseeformat
& \glsseeitemformat
\glsseeformat .............. 227, 334, 603
see also \glsseeitem,
\glsseeitemformat,
\glsseeep
& \glsseeep
\glsseeitem .............. 603
see also \glsseeformat
& \glsseeitemformat
\glsseeitemformat ........ 191, 603
see also \glsseeitemformat
& \glsseeitem
\glsseelastoxfordsep ........ 603
see also \glsseeformat
& \glsseeep
\glsseelastsep ............... 603
see also \glsseeformat
& \glsseeep
\glssee list ............... 183, 603
\glssee ep ..................... 603
see also \glsseeformat
& \glssee lastsep
\glssetcategoryattribute .... 16, 153, 603
\glssetexpandfield .......... 45, 377
\glssetnexpandingfield .... 377, 378
\glssetwidest ............... 124, 365, 366, 603
\glsshortaccessdisplay ..... 604
\glsshortpluralaccessdisplay .. 604
\glsshorttok .................... 512, 604
\glsshowtarget ............. 604
\glsstartrange ........ .... 228, 604
\glsstepentry .............. 604
\glssubentrycounterlabel .... 604
\glssubentryitem ............ 604
\glssubgroupheading ......... 151
\glsymbol .................... 546, 604
\glsymbolaccessdisplay ...... 604
\glsymbolpluralaccessdisplay .. 604
\glsymbolpluralaccessdisplay .. 604
\glsymbolsgroupname ....... 343, 349, 543, 604
\glslanguage .................. 475, 604
Index

\textbf{\texttt{\textbackslash Glstext}} 515, 605
\textbf{\texttt{\textbackslash glstext}} 162, 217, 515, 605
\textbf{\texttt{\textbackslash glstextaccessdisplay}} 605
\textbf{\texttt{\textbackslash glstextformat}} 520, 605
\textbf{\texttt{\textbackslash glstextup}} 605
\textbf{\texttt{\textbackslash glstildechar}} 23, 277, 605
\textbf{\texttt{\textbackslash glstreedefaultnamenfmt}} 605
\textbf{\texttt{\textbackslash glstreegroupheaderfmt}} 462, 605
\textbf{\texttt{\textbackslash glstreenamefmt}} 462, 605
\textbf{\texttt{\textbackslash glstreenamefmt}} 605
\textbf{\texttt{\textbackslash glstreennonamedesc}} 605
\textbf{\texttt{\textbackslash glstreepredesc}} 605
\textbf{\texttt{\textbackslash glstreeprelocation}} 605
\textbf{\texttt{\textbackslash glstriggerrecordformat}} 42, 172, 230, 231, 605
\textbf{\texttt{\textbackslash glsunet}} 605
\textbf{\texttt{\textbackslash glsupdatewidest}} 124, 365, 366, 605
\textbf{\texttt{\textbackslash glsupdatewidest}} 605
\textbf{\texttt{\textbackslash glsupercase}} 370, 514, 606
\textbf{\texttt{\textbackslash glsusabrvfont}} 324, 606
\textbf{\texttt{\textbackslash glsuselongfont}} 328, 606
\textbf{\texttt{\textbackslash glsuserdescription}} 313, 606
\textbf{\texttt{\textbackslash glsuseri}} 606
\textbf{\texttt{\textbackslash glsuserii}} 606
\textbf{\texttt{\textbackslash glsuseriii}} 606
\textbf{\texttt{\textbackslash glsuseriv}} 606
\textbf{\texttt{\textbackslash glsuserv}} 606
\textbf{\texttt{\textbackslash glsuservi}} 606
\textbf{\texttt{\textbackslash glsxtrrecord}} 230
\textbf{\texttt{\textbackslash glsxtrrecordnameref}} 40
\textbf{\texttt{\textbackslash glsxtrresource}} 18, 19
\textbf{\texttt{\textbackslash glsxtrwr glossarylocation}} 251
\textbf{\texttt{\textbackslash glsxtrabrvfont}} 607
\textbf{\texttt{\textbackslash glsxtrabrvfootnote}} 607
\textbf{\texttt{\textbackslash glsxtrabrvplursuffix}} 226, 607
\textbf{\texttt{\textbackslash glsxtrabrvtype}} 321, 373, 607
\textbf{\texttt{\textbackslash glsxtraddlabelprefix}} 82, 87, 607
\textbf{\texttt{\textbackslash glsxtralttreectext}} 515, 605
\textbf{\texttt{\textbackslash glsxtralttreeSymbolDescLocation}} 528, 541, 607
\textbf{\texttt{\textbackslash glsxtrapptocsvfield}} 183, 607
\textbf{\texttt{\textbackslash GlXtrAutoAddOnFormat}} 236–239, 607
\textbf{\texttt{\textbackslash glsxtrautoindexassignsort}} 17, 607
\textbf{\texttt{\textbackslash glsxtrautoindexentry}} 17, 607
\textbf{\texttt{\textbackslash GlXtrBibTeXEntryAliases}} 15, 71, 97, 99, 413, 496, 607
\textbf{\texttt{\textbackslash glsxtrbookindexname}} 465, 530, 545, 607
\textbf{\texttt{\textbackslash glsxtrbookindexprelocation}} 233, 608
\textbf{\texttt{\textbackslash glsxtrclearlabelprefixes}} 608
\textbf{\texttt{\textbackslash glsxtrclearlabelprefixes}} 608
\textbf{\texttt{\textbackslash glsxtrprependlabelprefixes}} 608
\textbf{\texttt{\textbackslash glsxtrcombiningdiacriticalrules}} 278, 608
\textbf{\texttt{\textbackslash glsxtrcontrolrules}} 278, 608
\textbf{\texttt{\textbackslash glsxtrcopytoglossary}} 128, 129, 364
\textbf{\texttt{\textbackslash GlXtrDefaultResourceOptions}} 111
\textbf{\texttt{\textbackslash glsxtrdetoklocation}} 365, 608
\textbf{\texttt{\textbackslash glsxtdigitrules}} 278, 608
\textbf{\texttt{\textbackslash glsxtrdisplaylocnameref}} 41
\textbf{\texttt{\textbackslash glsxtrdisplaysupplloc}} 255
\textbf{\texttt{\textbackslash GlXtrDualBackLink}} 313
\textbf{\texttt{\textbackslash GlXtrDualField}} 305, 313
\textbf{\texttt{\textbackslash glsxtremsuffix}} 608
\textbf{\texttt{\textbackslash GlXtrEnableInitialTagging}} 23, 435, 512, 608
\textbf{\texttt{\textbackslash glsxtenablerecordcount}} 42, 609
\textbf{\texttt{\textbackslash glsxtrencountfor}} 519, 520, 609
\textbf{\texttt{\textbackslash glsxtrforcsvfield}} 609
\texttt{& listbreak}
\textbf{\texttt{\textbackslash glsxtreentryfmt}} 62, 519, 520
\textbf{\texttt{\textbackslash glsxtrentreyparentname}} 609
\textbf{\texttt{\textbackslash GlXtrExpandedFmt}} 609
\textbf{\texttt{\textbackslash glsxtrfielddolistloop}} 98, 228, 609
\texttt{& listbreak}
\texttt{& listbreak}
\textbf{\texttt{\textbackslash glsxtrfieldforlistloop}} 98, 100, 138, 228, 609
\texttt{& listbreak}
\textbf{\texttt{\textbackslash glsxtrfielddolistloop}} 609
\textbf{\texttt{\textbackslash glsxtrfieldforlistloop}} 609
\textbf{\texttt{\textbackslash glsxtrfieldforlistloop}} 609

657
Index

\texttt{see also \textbackslash glsxtrfieldforlistloop,}
\texttt{\textbackslash glsxtrfielddolistloop,}
\texttt{\textbackslash DTLformatlist}
\texttt{& \textbackslash glsxtrfieldformatlist}
\texttt{\glsxtrfieldforlistloop 609}
\texttt{\glsxtrfieldformatlist 609}
\texttt{see also \textbackslash glsxtrfieldforlistloop,}
\texttt{\textbackslash glsxtrfielddolistloop,}
\texttt{\textbackslash DTLformatlist}
\texttt{& \textbackslash glsxtrfieldformatcvslist}
\texttt{\glsxtrfieldforlistloop 609}
\texttt{\glsxtrfielddolistloop 609}
\texttt{\glsxtrfieldformatlist 609}
\texttt{\glsxtrfielddolist 319}
\texttt{\glsxtrfieldxifinlist 609}
\texttt{\glsxtrfieldxifinlist 609}
\texttt{\glsxtrfieldfinish 609}
\texttt{\glsxtrfieldfinish 609}
\texttt{\glsxtrfieldfinish 609}
\texttt{\glsxtrfmt 61–63, 427}
\texttt{\glsxtrfmt 61–63, 427}
\texttt{\glsxtrfmtdisplay 62}
\texttt{\glsxtrfmtdisplay 62}
\texttt{\textbackslash GlsXtrFmtDefaultOptions 62, 503, 609}
\texttt{\glsxtrfmtdisplay 610}
\texttt{\textbackslash GlsXtrFmtField 61, 503, 610}
\texttt{\textbackslash GlsXtrFootnotename 610}
\texttt{\textbackslash glsxtrforcsvfield 519, 610}
\texttt{see also \textbackslash glsxtrrendfor}
\texttt{\textbackslash GlsXtrForeignText 610}
\texttt{\textbackslash GlsXtrForUnsetBufferedList 610}
\texttt{\textbackslash glsxstractionrules 278, 610}
\texttt{\textbackslash glsxstrfull 610}
\texttt{\textbackslash glsxstrfullsep 610}
\texttt{\textbackslash glsxstrGeneralLatinIIrules 610}
\texttt{\textbackslash glsxstrGeneralLatinIrules 279, 611}
\texttt{\textbackslash glsxstrGeneralLatinIIrules 610}
\texttt{\textbackslash glsxstrGeneralLatinIrules 279, 611}
\texttt{\textbackslash glsxstrGeneralLatinIIrules}
\texttt{\textbackslash glsxstrGeneralLatinIIrules}
\texttt{\textbackslash glsxstrGeneralLatinIrules}
\texttt{\textbackslash glsxstrGeneralLatinIrules}
\texttt{see also \textbackslash glsxstrGeneralLatinIIrules,}
\texttt{\textbackslash glsxstrGeneralLatinIrules,}
\texttt{\textbackslash glsxstrGeneralLatinIrules,}
\texttt{\textbackslash glsxstrGeneralLatinIrules,}
\texttt{\glsxtrGeneralLatinIIrules}
\texttt{\glsxtrGeneralLatinIrules}
\texttt{\glsxtrGeneralLatinIrules}
\texttt{\glsxtrGeneralLatinIrules}
\texttt{& \glsxtrGeneralLatinIIrules}
\texttt{\glsxtrGeneralLatinIIrules 278, 611}
\texttt{\glsxtrGeneralLatinIrules 278, 611}
\texttt{\glsxtrGeneralLatinIIrules 611}
\texttt{\glsxtrGeneralLatinIrules 278, 611}
\texttt{\glsxtrGeneralLatinIIrules 611}
\texttt{\glsxtrGeneralLatinIrules 278, 611}
\texttt{\glsxtrglossentry 10, 201, 611}
\texttt{\glsxtrglossentryother 100, 129, 130, 182, 296,}
\texttt{486, 496, 611}
\texttt{\textbackslash GLSXTRhiernamename 23, 611}
\texttt{see also \textbackslash glsxtrhiernamename}
\texttt{& \textbackslash glsxtrhiernamename}
\texttt{\GlsXtrhiernamename 23, 611}
\texttt{see also \textbackslash glsxtrhiernamename}
\texttt{& \textbackslash glsxtrhiernamename}
\texttt{\GlsXtrhiernamename 23, 611}
\texttt{see also \textbackslash glsxtrhiernamename}
\texttt{& \textbackslash glsxtrhiernamename}
\texttt{\GlsXtrhiernamename 23, 190, 191, 612}
\texttt{see also \textbackslash glsxtrhiernamename}
\texttt{\GlsXtrhiernamename 23, 190, 191, 612}
\texttt{see also \textbackslash glsxtrhiernamename}
\texttt{\GlsXtrhiernamename 23, 190, 191, 612}
\texttt{see also \textbackslash glsxtrhiernamename}
\texttt{\textbackslash glsxtrhyphenrules 15, 278, 612}
\texttt{\glsxtrhyphensuffix 612}
\texttt{\glsxtrifcustomdiscardperiod 187}
\texttt{\textbackslash GlsXtrIfFieldCmpNum 264, 265, 612}
\texttt{\glsxtrifhasfield 140, 612}
\texttt{see also \textbackslash GlsXtrIfFieldNonZero}
\texttt{\GlsXtrIfFieldEqNum 140, 612}
\texttt{& \GlsXtrIfFieldEqNum}
\texttt{\textbackslash GlsXtrIfFieldEqStr 612}
\texttt{\GlsXtrIfFieldEqStr 612}
\texttt{\GlsXtrIfFieldNonZero 140, 613}
\texttt{see also \GlsXtrIfFieldEqStr}
\texttt{\GlsXtrIfFieldUnDef 187, 613}
\texttt{see also \ifglsfieldvoid}
\texttt{\glsxtrifhasfield 140, 187, 458, 497, 519,}
\texttt{531, 613}
\texttt{see also \GlsXtrIfFieldUnDef}
\texttt{\glsxtrifhasfield* 187, 613}
\texttt{see also \GlsXtrIfFieldUnDef}
\texttt{\GlsXtrIfHasNonZeroChildCount 138}
\texttt{see also \GlsXtrIfFieldNonZero}
\texttt{\glsxtrifhyphenstart 613}
\texttt{\GlsXtrIfInGlossary 364, 613}
Index

see also \GlsXtrSetDefaultNumberFormat
\GlsXtrSetDefaultNumberFormat \ 228, 229, 287, 450, 514, 618
\GlsXtrSetField \ 138, 165, 184, 251, 319, 619
\glsxtrsetglossarylabel \ 619
\glsxtrsetgroupstyle \ 5, 8, 31, 169, 343, 346
\glsxtrsetpopts \ 619
\GlsXtrSetRecordCountAttribute \ 42
\glsxtrSetWidest \ 125, 365, 366, 619
see also \GlsSetWidest
& \GlsxtrSetWidestFallback \ 5, 30, 150
custom \ see custom group
date \ see date group
datetime \ see date-time group
empty \ see empty group (unknown commands)
hierarchical \ see sub-group
letter \ see letter group
non-letter \ see non-letter group
number \ see number group
small \ see small group
symbol \ see symbol group
time \ see time group

H

handlers \ 230, 233, 497, 519, see also loops
\heartsuit \ 7, 622
hierarchical entry
hierarchical group \ see sub-group
homograph \ 147, 220, 363
\hyperbf \ 39, 40, 85, 228–231, 235, 236, 336, 531, 547, 622
\hyperemph \ 235, 237, 622
\hyperit \ 39, 40, 622
\hyperlink \ 251, 622
\hyperref \ 622
\hyperrefurl \ 252, 622
\hyperrm \ 236–238, 622
\hypsfr \ 39, 40, 622
Index

IETF (Internet Engineering Task Force) \ldots 18, 30, 265, 266, 518
\ifcase \ldots 246, 622
\ifsodef \ldots 339, 622
\ifcsstrequal \ldots 622
\ifcsstring \ldots 623
\ifdef \ldots 367, 623
\ifdefstrequal \ldots 623
\ifDTLlistskipempty \ldots 623
\IfFileExists \ldots 519, 623
\ifglossaryexists \ldots 11, 452, 623
\ifglossaryexists* \ldots 11, 364, 452, 623
\ifglsentryexists \ldots 128, 363, 623
\ifglsfieldcseq \ldots 623
\ifglsfielddefeq \ldots 623
\ifglsfieldeq \ldots 623
\ifglsfieldvoid \ldots 623
\seealso \GlsXtrIfFieldUndef
\ifglsfielddescchildren \ldots 138, 623
\ifglsfielddesc \ldots 521, 624
\seealso \ifgls hassym
\& \GlsXtrIfFieldUndef
\ifglsfieldhasfield \ldots 140, 187, 458, 624
\seealso \GlsXtrIfFieldUndef
\& \GlsXtrIfFieldUndef
\ifglsfieldaslong \ldots 624
\ifglsfieldastext \ldots 138, 624
\ifglsfieldasprefix \ldots 624
\seealso \ifglsfieldasc, \GlsXtrIfFieldUndef
\& \GlsXtrIfFieldUndef
\ifglsfieldasprefixfirstplural \ldots 624
\seealso \ifglsfieldasc, \GlsXtrIfFieldUndef
\& \GlsXtrIfFieldUndef
\ifglsfieldasprefixplural \ldots 624
\seealso \ifglsfieldasc, \GlsXtrIfFieldUndef
\& \GlsXtrIfFieldUndef
\ifglsfieldashort \ldots 518, 624
\ifglsfieldasuppressedesc \ldots 624

\seealso \ifglshasdesc
\ifglshassymbol \ldots 624
\seealso \ifglshasdesc, \GlsXtrIfFieldUndef
\& \GlsXtrIfFieldUndef
\ifglshasused \ldots 624
\seealso \GlsXtrIfUndefinedOrUndefined
\& \GlsXtrIfWasFirstUse
\ifglshasinsertinside \ldots 625
\ifignoredglossary \ldots 625
\ifnum \ldots 339, 625
\ifstrempty \ldots 198, 625

ignored glossary \ldots 10
ignored record \ldots 38, 42, 43, 52, 98, 172, 227–231, 287
\immediate \ldots 1, 625
\IN (quark) \ldots 16, 123
\index \ldots 16, 625
\indexname \ldots 625
\input \ldots 1, 52, 625
\INTERPRET (quark) \ldots 16, 117
interpret-preamble.bib \ldots 384, 385, 398, 404, 407, 428, 478, 482, 501, 538
interpret-preamble2.bib \ldots 384, 385, 404, 490, 492
\invfmt \ldots 427
iteration \ldots \seealso loops

\jobname \ldots 110, 132, 156, 625
JRE (Java Runtime Environment) \ldots 28, 273, 398, 399, 485
JVM (Java Virtual Machine) \ldots 35, 267, 273, 372

L
\L \ldots 209, 625
\l \ldots 209, 626
\label \ldots 250, 626
\seealso \ref
& \pageref
label prefixes
dual. \ldots 50, 79, 160–163, 301, 530
ext(\hat{a}) \ldots 50, 160–164, 407
tertiary. \ldots 96, 160, 162, 314
Index

\LABELIFY (quark) .......................... 117
\LABELIFYLIST (quark) .................. 117
language resource file (.xml)
  groupresource <lc> .......................... 32, 344
  sentence.terminators .................. 186, 187
tag.page .................................. 247, 338
tag.pages ................................ 247, 338
\LC (quark) ................................. 16, 117
\LEN (quark) ................................. 16, 117, 120–122, 148
\let ........................................ 487, 626
letter group ................................. 3, 6, 31, 32, 345, 346
link text ................................... 599
\listbreak .................................. 626
  see also \Vforlistloop,
  \glsxtrfieldforlistloop,
  \glsxtrfielddolistloop
& \glsxtrrendfor
\listxadd ................................ 519, 626
\loadglsentries .......................... 1, 52, 377, 626
locale
document ................................. Java
  resource ................................. 28, 265, 273, 398, 485
locale list ................................. 13, 41, 133–135, 147, 195, 227, 232,
  233, 301, 333, 449, 632
  ranges .................................. see range
\longnewglossaryentry ................. 88, 217, 310, 378
\longnewglossaryentry* ............... 320, 626
\longprovideglossaryentry ........... 379
longtable environment ................... 475
loops ...................................... 15, 21, 98, 138, 228, 519, 520
lower case ................................. 27, 117, 206–209, 267, 268, 280, 289,
  290, 294, 295, 304, 353, 370
\mainmatter .............................. 229, 626
\makefirstuc .............................. 13, 45, 208–210, 312, 371, 515,
  582, 627
\makeglossaries ......................... 4, 285, 538, 627
MakeLowercase .......................... 294, 627
\makenoidxglossaries .................. 4, 627
\makeTextLowercase ..................... 370, 627
\makeTextUppercase ..................... 370, 627
\MakeUppercase ......................... 21, 377, 627
markupalanguages.bib .................. 435, 512, 524
\mathcal ................................. 24, 627
\mathgreek.bib .......................... 414, 419, 499
\mathord .................................. 426, 627
\mathobjects.bib ......................... 427, 501
\mathrelations.bib ....................... 423, 425, 499
\MFUnocap .................................. 207, 627
\MFUskippunc ............................ 208, 209, 214, 627
\MFUwordbreak ......................... 207, 209, 215, 627
\mgls ..................................... 105–107, 316, 627
\MGP (quark) .............................. 16, 116, 159
\midrule ................................... 475, 627
minerals.bib ............................... 444, 449, 524
micsymbols.bib .......................... 431, 506, 509, 542, 543
\mtxfmt .................................. 427
\multiglossaryentry ........................ 57, 105–107, 315, 318,
  333, 628
\multiglossaryentrysetup .............. 106, 108, 628
\n (newline) .................................. 16, 628
\nary ...................................... 420
\newabbreviation ........................ 53, 74, 88, 112, 217, 321,
  322, 327, 373, 374, 380
\newcommand .............................. 86, 319, 376, 382, 383, 628
\newduality ............................... 87, 373, 374, 382, 383
\newentry .................................. 378, 628
\newglossary .............................. 10, 628
\newglossary* ............................. 10, 264, 470, 628
\newglossaryentry ........................ 45, 51, 110, 174, 217, 218,
  232, 283, 372, 375, 378
\newglossarystyle ....................... 475, 628
\newignoredglossary ..................... 10, 629
\newignoredglossary* ................... 10, 629
\newnum .................................. 381, 629
\newrobustcmd ......................... 347, 629
\newsym .................................. 381, 629
\newterm .................................. 373, 374, 379
\NG ......................................... 209, 629
\ng ........................................ 209, 629
\NIN (quark) .............................. 16, 123

\MC

662
Index

\nobreakspace 35, 36, 205, 629
\NoCaseChange 208, 209, 213, 514, 629
\NOCHANGE (quark) 118
non-ASCII 25, 50, 51, 256, 289, 437, 516, 518
\noldacronym 373, 374, 380, 630
\notprefixof (quark) 16, 123
\notsuffixof 16, 123
\NULL (quark) 16, 120, 618
\number 294, 630
\numbergroup 31, 264, 349, 350
\numspacefmt 427

O
\O 157, 209, 630
\o 209, 630
\OE 209, 213, 630
\oe 209, 213, 630
\oldacronym 373, 374, 380, 630
\see also \newacronym
\omicron 414, 415, 630
openin_any 17, 372
openout_any 17, 372

P
package options
    abbreviations 8, 74, 88, 561
    accsupp 54, 561
    acronym 561
    acronymlists 561
    acronyms 561, 570
    automake 1, 561
    autoseindex 561
    counter 251, 561
    counterwithin 561
    debug 561, 604
    docdef 562
    entrycounter 561, 562
    esclocations 562
    hyperfirst 562

index 379, 486, 525, 562
indexcounter 41, 250, 251, 562
indexcrossrefs 562
indexonlyfirst 562
makeindex 562
nogroups 3, 30, 562
nopath 30, 562
nomain 17, 372, 494, 525, 563
nomissingglstext 563
numberedsection 563, 564
numbers 381, 563
order 564
postdot 185, 408, 457, 486, 494, 564
postpunc 563, 564
record 4, 10, 15, 19, 30, 40, 41, 110, 114, 128, 133, 227, 237, 253, 261, 278, 319, 457, 486, 564
sanitizesort 564
savenumberlist 564
savewrites 564
section 472, 525, 564
seeaunumberlist 564
seenoindex 564
shortcuts 47, 565
sort 110, 565
style 457, 475, 565
stylemods 457, 462, 465, 469, 475, 486, 494, 499, 525, 565
subentrycounter 362, 565, 592
symbols 8, 86, 172, 381, 565
toc 563, 565

663
## Index

<p>| Post-name hook | 466, 479, 484, 486, 521, 546 |
| primary | 71 |
| \PREFIXOF (quark) | 16, 123 |
| \printglossaries | 4, 285, 631 |
| \printglossary | 4, 631 |
| \printnoidxglossaries | 631 |
| \printnoidxglossary | 631 |
| \printunsrtglossaries | 4, 5, 631 |
| \printunsrtglossary | 4, 30, 73, 112, 129, 231, 296, 297, 343, 619, 631 |
| entrycounter | 632 |
| groups | 453 |
| label | 619 |
| leveloffset | 453 |
| nogroupskip | 82, 632 |
| nonumberlist | 487, 632 |
| nopostdot | 487, 632 |
| numberedsection | 632 |
| prefix | 632 |
| style | 487, 632 |
| subentrycounter | 632 |
| target | 129, 297, 487 |
| targetnameprefix | 632 |
| title | 5, 10, 297, 487 |
| toctitle | 633 |
| type | 10, 487 |
| \provideignoredglossary* | 11, 48, 155, 172, 296, 633 |
| \providemultiglossaryentry | 105, 107, 318, 633 |
| \ProvidesPackage | 36, 634 |
| Q |
| quarks (bib2gls) | 15, 114–117, 120 |
| \ref | 209, 626, 634, see also \label |
| \refstepcounter | 41, 250, 634 |
| regexp (or regex) see regular expressions |
| range | 39, 44, 227–231, 235, 238–241, 244, 250, 251, 335, 336 |
| explicit | 38, 39, 44, 227–233, 240, 241, 244, 336 |
| implicit | 228, 229, 240, 241, 244, 249, 335, 336 |
| interloper | 229, 240, 336 |
| \resource options |
| abbreviation-name-fallback | 69, 83, 219, 324, 529 |
| abbreviation-sort-fallback | 69, 83, 88, 91, 94, 219, 223, 224, 275, 276, 466, 512, 527, 529 |
| abbreviation-text-fallback | 219 |
| action | 128–130, 134, 154, 182, 295, 296, 364, 486, 526 |
| adopted-parent-field | 101, 102, 105, 172 |
| alias | 133, 245 |
| alias-loc | 12, 246 |
| append-prefix-field | 204, 205 |
| append-prefix-field-cs | 205 |
| append-prefix-field-cs-exceptions | 205 |
| append-prefix-field-exceptions | 205 |
| append-prefix-field-nbsp-match | 205, 206 |</p>
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>assign-missing-field-action</td>
<td>120, 179, 181</td>
</tr>
<tr>
<td>assign-override</td>
<td>176, 179, 181</td>
</tr>
<tr>
<td>bibtext-contributor-fields</td>
<td>12, 197,</td>
</tr>
<tr>
<td></td>
<td>198, 367, 368, 385, 399, 404</td>
</tr>
<tr>
<td>bibtextentry-sort-fallback</td>
<td>97, 222,</td>
</tr>
<tr>
<td></td>
<td>275, 276</td>
</tr>
<tr>
<td>break-at</td>
<td>188, 191, 197, 225, 265, 267, 279–282, 299, 307</td>
</tr>
<tr>
<td>break-at-match</td>
<td>281, 299, 308</td>
</tr>
<tr>
<td>break-at-match-op</td>
<td>281, 299, 308</td>
</tr>
<tr>
<td>break-at-not-match</td>
<td>281, 299, 308</td>
</tr>
<tr>
<td>break-marker</td>
<td>280, 281, 299, 308</td>
</tr>
<tr>
<td>category</td>
<td>13, 67, 81, 97, 104, 128, 155, 156, 169–171, 258, 304, 315, 320, 321, 478, 484, 496, 539</td>
</tr>
<tr>
<td>charset</td>
<td>35, 51, 124, 187, 492</td>
</tr>
<tr>
<td>check-end-punctuation</td>
<td>13, 56, 186, 187, 492</td>
</tr>
<tr>
<td>combine-dual-locations</td>
<td>13, 82, 96, 301, 303</td>
</tr>
<tr>
<td>compact-ranges</td>
<td>244, 245, 335</td>
</tr>
<tr>
<td>compound-add-hierarchy</td>
<td>13, 315</td>
</tr>
<tr>
<td>compound-adjust-name</td>
<td>13, 316</td>
</tr>
<tr>
<td>compound-dependent</td>
<td>13, 315</td>
</tr>
<tr>
<td>compound-has-records</td>
<td>316</td>
</tr>
<tr>
<td>compound-main-type</td>
<td>317, 318</td>
</tr>
<tr>
<td>compound-options-global</td>
<td>315</td>
</tr>
<tr>
<td>compound-other-type</td>
<td>317, 318</td>
</tr>
<tr>
<td>compound-type-override</td>
<td>317, 318</td>
</tr>
<tr>
<td>compound-write-def</td>
<td>107, 315, 318</td>
</tr>
<tr>
<td>contributor-order</td>
<td>197, 368</td>
</tr>
<tr>
<td>copy-action-group-field</td>
<td>128, 129, 182, 486</td>
</tr>
<tr>
<td>copy-alias-to-see</td>
<td>13, 182</td>
</tr>
<tr>
<td>copy-to-glossary</td>
<td>115, 130, 131, 364</td>
</tr>
<tr>
<td>copy-to-glossary-missing-field action</td>
<td>130, 131</td>
</tr>
<tr>
<td>counter</td>
<td>13, 55, 182, 305</td>
</tr>
<tr>
<td>cs-label-prefix</td>
<td>162, 163, 530</td>
</tr>
<tr>
<td>custom-sort-fallbacks</td>
<td>67, 68, 218–224, 275, 276</td>
</tr>
<tr>
<td>date-field-format</td>
<td>203, 204, 306</td>
</tr>
<tr>
<td>date-field-locale</td>
<td>203, 204, 306</td>
</tr>
<tr>
<td>date-fields</td>
<td>13, 203, 204, 306</td>
</tr>
<tr>
<td>date-sort-format</td>
<td>203, 204, 272–274, 293, 300, 309, 399</td>
</tr>
<tr>
<td>date-sort-locale</td>
<td>204, 272–274, 292, 300, 309</td>
</tr>
<tr>
<td>date-time-field-format</td>
<td>203, 306</td>
</tr>
<tr>
<td>date-time-field-locale</td>
<td>203, 204, 306</td>
</tr>
<tr>
<td>date-time-field-locale</td>
<td>203, 204, 306</td>
</tr>
<tr>
<td>date-time-fields</td>
<td>13, 203, 204, 368</td>
</tr>
<tr>
<td>decomposition</td>
<td>289, 300, 308</td>
</tr>
<tr>
<td>dependency-fields</td>
<td>12, 14, 52, 133, 166</td>
</tr>
<tr>
<td>description-case-change</td>
<td>70, 216</td>
</tr>
<tr>
<td>dual-abbrv-backlink</td>
<td>91, 313, 314</td>
</tr>
<tr>
<td>dual-abbrv-map</td>
<td>93, 310</td>
</tr>
<tr>
<td>dual-abbrventry-backlink</td>
<td>313, 314</td>
</tr>
<tr>
<td>dual-abbrventry-map</td>
<td>311</td>
</tr>
<tr>
<td>dual-break-at</td>
<td>188, 279, 307, 529</td>
</tr>
<tr>
<td>dual-break-at-match</td>
<td>281, 308</td>
</tr>
<tr>
<td>dual-break-at-match-op</td>
<td>281, 308</td>
</tr>
<tr>
<td>dual-break-at-not-match</td>
<td>281, 308</td>
</tr>
<tr>
<td>dual-break-marker</td>
<td>281, 308</td>
</tr>
<tr>
<td>dual-category</td>
<td>13, 79, 170, 304, 315</td>
</tr>
<tr>
<td>dual-counter</td>
<td>13, 305</td>
</tr>
<tr>
<td>dual-date-field-format</td>
<td>203, 204, 306</td>
</tr>
<tr>
<td>dual-date-field-locale</td>
<td>203, 204, 306</td>
</tr>
<tr>
<td>dual-date-sort-format</td>
<td>309</td>
</tr>
<tr>
<td>dual-date-sort-locale</td>
<td>292, 309</td>
</tr>
<tr>
<td>dual-date-time-field-format</td>
<td>203, 306</td>
</tr>
<tr>
<td>dual-date-time-field-locale</td>
<td>203, 204, 306</td>
</tr>
<tr>
<td>dual-decomposition</td>
<td>289, 308</td>
</tr>
<tr>
<td>dual-entry-backlink</td>
<td>312–314</td>
</tr>
<tr>
<td>dual-entry-map</td>
<td>309–313</td>
</tr>
<tr>
<td>dual-entryabbrev-backlink</td>
<td>314</td>
</tr>
<tr>
<td>dual-field</td>
<td>55, 305, 313, 531</td>
</tr>
<tr>
<td>dual-group-formation</td>
<td>295, 309, 353</td>
</tr>
<tr>
<td>dual-identical-sort-action</td>
<td>282, 308</td>
</tr>
<tr>
<td>dual-indexabbrev-backlink</td>
<td>314</td>
</tr>
<tr>
<td>dual-indexabbrev-map</td>
<td>312</td>
</tr>
<tr>
<td>dual-indexentry-backlink</td>
<td>80, 314</td>
</tr>
<tr>
<td>dual-indexentry-map</td>
<td>311</td>
</tr>
<tr>
<td>dual-indexsymbol-backlink</td>
<td>314</td>
</tr>
</tbody>
</table>
Index

dual-indexsymbol-map ....... 311, 528
  dual-letter-number-punc-rule
  290, 309
  dual-letter-number-rule .... 289, 309
  dual-long-case-change ....... 215, 305
  dual-missing-sort-fallback 276, 307
  dual-numeric-locale ....... 292, 309
  dual-numeric-sort-pattern 292, 309
  dual-prefix ............ 50, 72, 79, 160–163, 301, 369, 530
  dual-short-case-change .... 215, 305
  dual-short-plural-suffix .... 227
dual-short-case-change ...... 72, 73, 97, 138, 188, 292, 301, 306, 307
dual-sort ........... 72, 275, 307
  dual-sort-number-pad ....... 282, 308, 529
  dual-sort-pad-minus ....... 282, 308
  dual-sort-pad-plus ....... 282, 308
dual-sort-replace ....... 277, 307
dual-sort-rule ........... 307
dual-sort-suffix ........... 283, 308
dual-sort-suffix-marker ....... 288, 308
dual-strength ........... 288, 308
dual-symbol-backlink ....... 313, 314
dual-symbol-map ........... 311
dual-time-field-format ....... 203, 204, 306
dual-trim-sort ........... 276, 307
dual-type ........... 13, 32, 79, 86, 88, 126, 128, 171, 172, 247, 248, 303, 305, 314, 528
duplicate-label-suffix ....... 161, 542
encapsulate-fields ........... 12, 198, 199
capsulate-fields* ........... 12, 199
capsulate-sort ........... 288
ten-sort-fallback ....... 66–68, 218–221, 225, 275, 276, 453
ext-prefixes ....... 50, 160–163, 369
field-aliases ........... 12, 31, 99, 101, 173, 393–399, 404, 407, 482, 490, 500
field-case-change ....... 12, 13, 200, 216
field-concat-sep ....... 16, 219–225
flatten ........... 128, 140, 143, 149, 173, 261, 274
flatten-lonely ....... 140–148, 362, 363
flatten-lonely-condition ....... 147, 148
flatten-lonely-missing-field
  -action ............... 148
flatten-lonely-rule ....... 140, 141, 144, 147, 148
format-decimal-fields ....... 199, 200
format-integer-fields ....... 165, 166, 199, 200
  group ........ 5, 12, 30, 31, 34, 52, 56, 128, 150, 169, 295, 467, 469
  group-formation ....... 150, 169, 295, 301, 309, 344, 352, 353
  group-level ....... 30, 150–152, 169, 345, 355
duplicate-label-suffix ....... 161, 542
hex-unicode-fields ....... 13, 202, 342
identical-sort-action ....... 165, 166, 264, 282–284, 287, 300, 308, 453, 457, 478, 506
ignore-fields ....... 165, 295, 307
long-case-change ....... 215, 305
long-case-change* ....... 215, 305
  letter-number-punc-rule ....... 269, 270, 290, 300, 309
letter-number-rule ....... 269–271, 289, 291, 300, 309
limit ............... 13, 14, 137, 138
loc-counters ....... 238, 240, 249, 250, 338, 339
loc-prefix ....... 18, 233, 246–248, 336–338
loc-prefix-def ....... 247, 248, 337
loc-suffix ........... 248, 338, 514
loc-suffix-def ....... 248, 338
target ........... 30, 124, 260, 267, 292
long-case-change ....... 215, 305
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>11, 14, 49, 129, 154, 155</td>
</tr>
<tr>
<td>master-resources</td>
<td>156</td>
</tr>
<tr>
<td>match</td>
<td>13, 119, 136, 137, 194, 281, 298, 482</td>
</tr>
<tr>
<td>match-action</td>
<td>137, 298</td>
</tr>
<tr>
<td>match-op</td>
<td>136, 137, 298</td>
</tr>
<tr>
<td>max-loc-diff</td>
<td>243, 244, 335</td>
</tr>
<tr>
<td>merge-ranges</td>
<td>38, 39, 44, 228, 229, 240, 241, 244, 336</td>
</tr>
<tr>
<td>merge-small-groups</td>
<td>30, 151, 152, 354</td>
</tr>
<tr>
<td>min-loc-range</td>
<td>228, 229, 240, 241</td>
</tr>
<tr>
<td>missing-parent-category</td>
<td>149, 150</td>
</tr>
<tr>
<td>missing-parents</td>
<td>149, 150</td>
</tr>
<tr>
<td>missing-sort-fallback</td>
<td>216, 219–222, 225, 275, 276, 298, 307</td>
</tr>
<tr>
<td>name-case-change</td>
<td>13, 128, 208, 212–215, 317, 514</td>
</tr>
<tr>
<td>no-case-change-cs</td>
<td>209, 214</td>
</tr>
<tr>
<td>not-match</td>
<td>137, 298</td>
</tr>
<tr>
<td>numeric-locale</td>
<td>272, 292, 300, 309</td>
</tr>
<tr>
<td>numeric-sort-pattern</td>
<td>272, 292, 300, 309</td>
</tr>
<tr>
<td>post-description-dot</td>
<td>185, 186</td>
</tr>
<tr>
<td>prefix-fields</td>
<td>204, 205</td>
</tr>
<tr>
<td>prefix-only-existing</td>
<td>165</td>
</tr>
<tr>
<td>primary-dual-dependency</td>
<td>301</td>
</tr>
<tr>
<td>primary-loc-counters</td>
<td>238, 240</td>
</tr>
<tr>
<td>primary-location-formats</td>
<td>233</td>
</tr>
<tr>
<td>principal-loc-counters</td>
<td>233, 238</td>
</tr>
<tr>
<td>principal-location-formats</td>
<td>232–235</td>
</tr>
<tr>
<td>progenitor-type</td>
<td>101, 172</td>
</tr>
<tr>
<td>progeny-type</td>
<td>101, 172</td>
</tr>
<tr>
<td>prune-iterations</td>
<td>196</td>
</tr>
<tr>
<td>prune-see-match</td>
<td>193–196</td>
</tr>
<tr>
<td>prune-see-op</td>
<td>194, 196</td>
</tr>
<tr>
<td>prune-see-op</td>
<td>196</td>
</tr>
<tr>
<td>prune-seealso-match</td>
<td>193–196</td>
</tr>
<tr>
<td>prune-xr</td>
<td>193, 195</td>
</tr>
<tr>
<td>record-label-prefix</td>
<td>162, 535</td>
</tr>
<tr>
<td>replicate-fields</td>
<td>12, 70, 101, 158, 173–177, 218, 399, 465, 484, 492, 500, 527</td>
</tr>
<tr>
<td>replicate-missing-field-action</td>
<td>174, 176</td>
</tr>
<tr>
<td>replicate-override</td>
<td>174–176, 399</td>
</tr>
<tr>
<td>save-child-count</td>
<td>138, 140</td>
</tr>
<tr>
<td>save-crossref-tail</td>
<td>183, 184</td>
</tr>
<tr>
<td>save-definition-index</td>
<td>165, 199, 371</td>
</tr>
<tr>
<td>save-from-alias</td>
<td>183</td>
</tr>
<tr>
<td>save-from-see</td>
<td>183</td>
</tr>
<tr>
<td>save-from-seealso</td>
<td>183</td>
</tr>
<tr>
<td>save-index-counter</td>
<td>41, 250–253</td>
</tr>
<tr>
<td>save-locations</td>
<td>4, 128, 227, 231, 232, 250, 480, 503</td>
</tr>
<tr>
<td>save-loclist</td>
<td>232, 233, 319</td>
</tr>
<tr>
<td>save-original-entrytype</td>
<td>12, 126, 170, 171, 184</td>
</tr>
<tr>
<td>save-original-entrytype-action</td>
<td>170, 171, 184</td>
</tr>
<tr>
<td>save-original-id</td>
<td>12, 165</td>
</tr>
<tr>
<td>save-original-id-action</td>
<td>165</td>
</tr>
<tr>
<td>save-primary-locations</td>
<td>232</td>
</tr>
<tr>
<td>save-root-ancestor</td>
<td>140</td>
</tr>
<tr>
<td>save-sibling-count</td>
<td>140</td>
</tr>
<tr>
<td>save-use-index</td>
<td>166, 200, 371</td>
</tr>
<tr>
<td>secondary-break-at</td>
<td>279, 299</td>
</tr>
<tr>
<td>secondary-break-at-match</td>
<td>281, 299</td>
</tr>
<tr>
<td>secondary-break-at-match-op</td>
<td>281, 299</td>
</tr>
<tr>
<td>secondary-break-at-not-match</td>
<td>281, 299</td>
</tr>
<tr>
<td>secondary-break-marker</td>
<td>281, 299</td>
</tr>
<tr>
<td>secondary-date-sort-format</td>
<td>300</td>
</tr>
<tr>
<td>secondary-date-sort-localize</td>
<td>292, 300</td>
</tr>
<tr>
<td>secondary-decomposition</td>
<td>289, 300</td>
</tr>
<tr>
<td>secondary-group-formation</td>
<td>295, 301, 353</td>
</tr>
<tr>
<td>secondary-identical-sort-action</td>
<td>282, 300</td>
</tr>
<tr>
<td>secondary-letter-number-punc-rule</td>
<td>289, 300</td>
</tr>
<tr>
<td>secondary-letter-number-rule</td>
<td>289, 300</td>
</tr>
<tr>
<td>secondary-match</td>
<td>298</td>
</tr>
<tr>
<td>secondary-match-action</td>
<td>298</td>
</tr>
<tr>
<td>secondary-match-op</td>
<td>298</td>
</tr>
<tr>
<td>secondary-missing-sort-fallback</td>
<td></td>
</tr>
</tbody>
</table>
Index

276, 298

secondary-not-match ............... 298
secondary-numeric-locale .......... 292, 300
secondary-numeric-sort-pattern
292, 300

secondary-sort-number-pad ...... 282, 299
secondary-sort-pad-minus ...... 282, 299
secondary-sort-pad-plus ...... 282, 299
secondary-sort-replace ...... 277, 299
secondary-sort-rule ...... 296, 299
secondary-sort-suffix ...... 283, 300
secondary-sort-suffix-marker
288, 300
secondary-strength .............. 288, 300
secondary-trim-sort .............. 276, 299
see .................................. 245, 333
seealso ................................ 52, 245
selection ......................... 1, 13, 14, 38, 52, 72, 98, 128, 133–135, 140, 173, 183, 194–196, 229, 296, 449, 469, 480, 482, 486, 503, 514, 543
short-case-change ................. 12, 69, 154, 213–215, 305
short-plural-suffix ............... 12, 227
shuffle ................................ 138, 261, 274
sort-label-list ................. 188, 193, 259
sort-number-pad ............... 268, 276, 282, 288, 299, 308
sort-pad-minus ............... 282, 299, 308
sort-pad-plus ............... 282, 299, 308
sort-replace ............... 159, 276, 280, 299, 307
sort-rule ...... 21, 22, 262, 266, 277, 299, 307
sort-suffix ............... 283, 288, 300, 308, 479, 500, 506
sort-suffix-marker ........ 284, 288, 300, 308
src 14, 19, 111, 119, 132, 134, 164, 253, 257
strength .................. 288, 289, 300, 308
strip-missing-parents ....... 148, 149
strip-trailing-nopost ........... 13, 186, 408
suffixF ........................ 240, 244
suffixFF ....................... 240, 244
supplemental-category ........... 256–258
supplemental-locations ........... 255, 341
supplemental-selection ........... 257
symbol-sort-fallback ........... 275, 276, 390, 457, 475, 529
tertiary-category ............... 96, 315
tertiary-prefix ................ 96, 162, 314, 369
tertiary-type ................... 96, 128, 172, 314
time-field-format ............... 203, 204, 306
time-field-locale ............... 203, 204, 306
time-fields ................. 13, 203, 204, 368
trigger-type ................. 11, 42, 49, 172
trim-sort ..................... 276, 299, 307
unknown-entry-alias .......... 97, 127
word-boundaries ............... 207, 215
write-preamble ................ 22, 124, 260, 385, 492
\rgls ...................... 42, 43, 172, 230
\rglsformat ................... 42
S

sample-authors.tex .......... 150, 490
sample-bacteria.tex .......... 463, 479, 526
sample-chemical.tex .......... 460, 529
sample-citations.tex ........... 494
sample-constants.tex ........... 455
sample-dual.tex ............... 382
sample-hierarchical.tex ........ 449–453
sample-markuplanguages.tex .... 512, 526
sample-maths.tex ............... 428, 501
sample-media.tex .............. 384, 408, 478, 492, 541
sample-msymbols.tex .......... 499–503

669
Index

sample-multi1.tex  16, 524, 535, 540, 545, 546
sample-multi2.tex  16, 535
sample-nested.tex  10, 134, 450
sample-people.tex  482, 492, 538, 539, 545
sample-textsymbols.tex  5, 31, 431, 500, 506, 509
sample-textsymbols2.tex  5, 31, 500, 508, 509
sample-units1.tex  467, 470
sample-units2.tex  470
sample-units3.tex  473, 529
sample-usergroups.tex  516
secondary  95, see also dual
section counter  254
\section  41, 634
\section*  472, 525, 634
\selectlanguage  52, 376, 377, 634
\selectlanguage*  192, 634
sentence case  117, 208
\setabbreviationstyle  69, 112, 634
\setcardfmt  427
\setcontentsfmt  427
\setentrycounter  257, 634
\setfmt  427
\setglossarypreamble  635
\setglossarypreamble  see also \glossarypreamble
\setglossarystyle  475, 635
\setmainlanguage  518, 635
\setmembershipfmt  427
\setmembershiponeargfmt  427
\setotherlanguage  520, 635
\setupglossaries  635
\showglocounter  635
\showglodesc  635
\showglodescplural  635
\showglofield  635
\showglofirst  635
\showglofirstpl  635
\showgloflag  635
\showglloglossaries  635
\showglolevel  636
\showgliloclist  636
\showglolong  636
\showgloname  636
\showgloparent  636
\showgloliteral  636
\showgloflag  636
\showglossarycounter  636
\showglossaryentries  636
\showglossarytitle  636
\showglosymbol  636
\showglosymbolplural  636
\showgloleft  636
\showglostext  636
\showgloleft  636
\showgloleft  636
\showgloquent  637
\showgloquenti  637
\showgloquentii  637
\showgloquentiii  637
\showgloquentiv  637
\showgloquentv  637
\showgloquentvi  637
\si  22, 209, 395, 475, 637
\si  395, 475, 541
\sigma  200, 637
\smallgroup  30, 151
\sortart  384, 385
\sortmediacreator  385, 492
\sortname  384, 385, 485, 545
\sorttop  259, 260, 384
\sortvonneame  384, 385, 399, 485, 545
\space  204–206, 215, 637
\SS  209, 637
\ss  209, 637
StackOverflowError  21
\string  15, 115, 159, 205, 225, 277, 288, 292, 293
\string  293
\struck  112–114
\sub-group  30, 150, 151
\subglossentry  230, 476, 637
\SUFFIXOF  (quark)  16, 123
\SUFFIXOF  (quark)  16, 123
\surd  386, 638
\switches  see command line options (bib2gls)
\switches & command line options (convertgls2bib)
\symbol  342, 638
\symbolgroup  see non-letter group

670
**Index**

<table>
<thead>
<tr>
<th>T</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\tableofcontents</td>
<td>201, 638</td>
</tr>
<tr>
<td>\tabularnewline</td>
<td>475, 638</td>
</tr>
<tr>
<td>tabularx environment</td>
<td>601</td>
</tr>
<tr>
<td>terms.bib</td>
<td>448, 449, 452, 543–545</td>
</tr>
<tr>
<td>tertiary</td>
<td>95</td>
</tr>
<tr>
<td>texoraudfstring</td>
<td>201, 361, 638</td>
</tr>
<tr>
<td>texparserlib.jar</td>
<td>20–22, 25, 29, 37</td>
</tr>
<tr>
<td>\textbf</td>
<td>228, 229, 242, 496, 638</td>
</tr>
<tr>
<td>\textcolor</td>
<td></td>
</tr>
<tr>
<td>\textstyle</td>
<td>214, 638</td>
</tr>
<tr>
<td>\textt</td>
<td>82, 212, 638</td>
</tr>
<tr>
<td>\textweathersymbol</td>
<td>520, 638</td>
</tr>
<tr>
<td>\textsc</td>
<td>479, 638</td>
</tr>
<tr>
<td>\textsuperscript</td>
<td>587, 638</td>
</tr>
<tr>
<td>\textsubscript</td>
<td>419, 638</td>
</tr>
<tr>
<td>\textsubscript</td>
<td>37, 291, 638</td>
</tr>
<tr>
<td>\textsuperscript</td>
<td>284, 638</td>
</tr>
<tr>
<td>\texttt</td>
<td></td>
</tr>
<tr>
<td>\textweathersymbol</td>
<td>432, 638</td>
</tr>
<tr>
<td>\textbf</td>
<td>209, 639</td>
</tr>
<tr>
<td>\textcolor</td>
<td>209, 639</td>
</tr>
<tr>
<td>\textstyle</td>
<td>512, 639</td>
</tr>
<tr>
<td>theglossary environment</td>
<td>633</td>
</tr>
<tr>
<td>theglossaryentry</td>
<td>639</td>
</tr>
<tr>
<td>theHentrycounter</td>
<td>40, 639</td>
</tr>
<tr>
<td>theHglossaryentry</td>
<td>639</td>
</tr>
<tr>
<td>time group</td>
<td>31, 351, 352</td>
</tr>
<tr>
<td>\TITLE (quark)</td>
<td>16, 118</td>
</tr>
<tr>
<td>title case</td>
<td>118, 207–209, 214, 216, 371</td>
</tr>
<tr>
<td>ROC (table of contents)</td>
<td>19</td>
</tr>
<tr>
<td>topics.bib</td>
<td>449, 509</td>
</tr>
<tr>
<td>toprow</td>
<td>475, 639</td>
</tr>
<tr>
<td>\TrackedLanguageFromDialect</td>
<td>519, 639</td>
</tr>
<tr>
<td>\TrackLangLastTrackedDialect</td>
<td>519, 639</td>
</tr>
<tr>
<td>TrackLocale</td>
<td>519, 639</td>
</tr>
<tr>
<td>transposefmt</td>
<td>427</td>
</tr>
<tr>
<td>trigraph</td>
<td>32</td>
</tr>
<tr>
<td>\TRIM (quark)</td>
<td>16, 117</td>
</tr>
<tr>
<td>TUG (\TeX Users Group)</td>
<td>437</td>
</tr>
</tbody>
</table>

| \u (accent)                                                     | 277, 639                                                        |
| \u (hex code quark)                                            | 16, 205, 224, 225, 277, 288, 292, 293                          |
| \UC (quark)                                                    | 16, 117, 178                                                   |
| unaryoperators.bib                                             | 426, 499                                                       |
| \textbf                                                        | 506, 640                                                        |
| \underline                                                      | 514, 640                                                        |
| unexpanded                                                      | 346, 347, 640                                                   |
| Unicode categories                                             |                                                                 |
| Letter, Lowercase                                              | 270                                                             |
| Letter, Modifier                                               | 270                                                             |
| Letter, Other                                                  | 270                                                             |
| Letter, Titlecase                                              | 270                                                             |
| Letter, Uppercase                                              | 270                                                             |
| Number, Decimal Digit                                          | 241, 268, 291                                                  |
| Punctuation, Close                                             | 185, 186                                                       |
| Punctuation, Dash                                              | 215                                                             |
| Punctuation, Final quote                                       | 185, 186                                                       |
| Punctuation, Other                                             | 185–187                                                        |
| Separator, Space                                               | 269                                                             |
| usepackage                                                      | 54, 376, 563, 640                                               |
| usergroups.bib                                                  | 437, 516                                                        |

| V                                                                 |                                                                 |
| \textbf                                                        | 25, 640                                                         |
| vec                                                             | 427                                                             |
| vegetables.bib                                                  | 446, 449, 524                                                  |
| \textbf                                                        | 26, 61, 427, 640                                               |

| W                                                                 |                                                                 |
| wrglossary counter                                              | 41, 56, 250–252                                                |
| \textbf                                                        | 640                                                             |
| \textbf                                                        | 99                                                              |
| see also \glsaccsupp                                            | 640                                                             |
| \xGlsXtrSetField                                                | 519, 640                                                        |
| \xifinlist                                                      | 519, 640                                                        |
| \xmakefirststuc                                                 | 168, 640                                                        |
| XML resource file                                               | see language resource file (.xml)                              |

671