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. 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].

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.

1The 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.
# Contents

## 1 Introduction
- 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 Indexing .................................................. 15
- 1.6 Security ................................................... 16
- 1.7 Localisation .............................................. 16
- 1.8 Conditional Document Build .............................. 17
- 1.9 Manual Installation ..................................... 18

## 2 TeX Parser Library .............................................. 20

## 3 Command Line Options ........................................... 28
- --help (or -h) ............................................... 28
- --version (or -v) .......................................... 28
- --debug [⟨n⟩] ................................................ 28
- --no-debug (or --nodebug) ................................ 28
- --verbose ..................................................... 28
- --no-verbose (or --noverbose) .............................. 29
- --silent ....................................................... 29
- --locale ⟨lang⟩ (or -l ⟨lang⟩) .............................. 29
- --log-file ⟨filename⟩ (or -t ⟨filename⟩) .................. 29
- --dir ⟨dirname⟩ (or -d ⟨dirname⟩) ......................... 29
- --interpret ................................................... 30
- --no-interpret .............................................. 30
- --no-break-space ......................................... 30
- --break-space ............................................... 30
- --cite-as-record .......................................... 31
- --no-cite-as-record ....................................... 31
- --warn-non-bib-fields ..................................... 31
- --no-warn-non-bib-fields ................................ 31
- --warn-unknown-entry-types ............................... 31
- --no-warn-unknown-entry-types ............................ 31
- --merge-wrglossary-records ............................... 32
- --no-merge-wrglossary-records .......................... 32
- --merge-nameref-on ⟨rule⟩ ................................. 32
Contents

--force-cross-resource-refs (or -x) ........................................ 33
--no-force-cross-resource-refs ............................................. 33
--support-unicode-script .................................................. 33
--no-support-unicode-script .............................................. 34
--list-known-packages ....................................................... 34
--packages ⟨list⟩ (or -p ⟨list⟩) ........................................ 35
--custom-packages ⟨list⟩ ..................................................... 35
--ignore-packages ⟨list⟩ (or -k ⟨list⟩) .................................. 35
--mfirstuc-protection ⟨list⟩|all (or -u ⟨list⟩|all) .................. 35
--no-mfirstuc-protection .................................................. 36
--mfirstuc-math-protection ............................................... 36
--no-mfirstuc-math-protection .......................................... 36
--nested-link-check ⟨list⟩|none ......................................... 36
--no-nested-link-check .................................................... 37
--shortcuts ⟨value⟩ .......................................................... 37
--map-format ⟨map:value list⟩ (or -m ⟨map:value list⟩) .......... 37
--group (or -g) .............................................................. 38
--no-group ..................................................................... 42
--tex-encoding ⟨name⟩ ....................................................... 42
--no-expand-fields .......................................................... 43
--expand-fields .............................................................. 43
--trim-fields ................................................................. 43
--trim-only-fields ⟨list⟩ .................................................... 43
--trim-except-fields ⟨list⟩ ................................................. 44
--no-trim-fields ............................................................. 44
--provide-glossaries ........................................................ 44
--record-count (or -c) ...................................................... 45
--no-record-count ........................................................... 45
--record-count-unit (or -n) ............................................... 46
--no-record-count-unit ..................................................... 46

4 .bib Format ........................................................................ 47
4.1 Encoding ................................................................. 47
4.2 Comments ............................................................... 48
4.3 Fields .................................................................... 48
4.4 Standard Entry Types .................................................. 55
   @string ................................................................. 55
   @preamble ............................................................... 55
4.5 Single Entry Types ..................................................... 60
   @entry ................................................................. 60
   @symbol ............................................................... 61
   @number .............................................................. 62
   @index .................................................................. 62
Contents

not-match={key=value list} ........................................ 116
match-action={value} ........................................... 116
limit={number} .................................................. 116

5.3 Hierarchical Options ........................................ 117
save-child-count={boolean} ................................. 117
save-sibling-count={boolean} ............................ 119
flatten={boolean} ............................................... 119
flatten-lonely={value} ....................................... 119
flatten-lonely-rule={value} .............................. 126
strip-missing-parents={boolean} ......................... 126
missing-parents={value} .................................... 126
missing-parent-category={value} ......................... 128

5.4 Master Documents ........................................ 128
master={name} .................................................. 130
master-resources={list} ..................................... 132

5.5 Field and Label Options ................................ 132
Label Options .................................................. 132
interpret-label-fields={boolean} ......................... 132
labelify={list} ................................................... 133
labelify-list={list} ............................................. 134
labelify-replace={list} .................................... 134
label-prefix={tag} ............................................. 136
duplicate-label-suffix={value} ......................... 137
record-label-prefix={tag} ................................ 137
cs-label-prefix={tag} ...................................... 137
ext-prefixes={list} ............................................ 138
save-original-id={value} .................................. 140
save-original-id-action={value} ......................... 140
dependency-fields={list} ................................ 141
Assignments ..................................................... 143
group={label} ................................................... 143
category={value} ............................................... 144
type={value} .................................................... 145
trigger-type={type} ........................................... 146
progenitor-type={type} ..................................... 147
progeny-type={type} ......................................... 147
adopted-parent-field={type} .............................. 147
abbreviation-name-fallback={field} .................. 147
ignore-fields={list} .......................................... 147
field-aliases={key=value list} .......................... 148
replicate-fields={key=value list} ...................... 149
replicate-override={boolean} .......................... 150
replicate-missing-field-action={value} ........... 150
counter={value} ................................................ 151
5.7 Location List Options ............................................ 181
save-locations=\{boolean\} ................................. 184
save-loclist=\{boolean\} ................................ 185
save-primary-locations=\{value\} ....................... 185
primary-location-formats=\{list\} ...................... 186
min-loc-range=\{value\} .................................. 190
max-loc-diff=\{value\} .................................. 193
suffixF=\{value\} ..................................... 194
suffixFF=\{value\} ..................................... 194
compact-ranges=\{value\} ............................... 194
see=\{value\} ........................................ 195
seealso=\{value\} ................................... 195
alias=\{value\} ....................................... 195
alias-loc=\{value\} .................................. 195
loc-prefix=\{value\} .................................. 196
loc-suffix=\{value\} .................................. 196
loc-counters=\{list\} ................................ 197
save-index-counter=\{value\} ......................... 199
5.8 Supplemental Locations .................................. 202
supplemental-locations=\{basename\} ................. 203
supplemental-selection=\{value\} ...................... 206
supplemental-category=\{value\} ...................... 207
5.9 Sorting ...................................................... 207
sort=\{value\} .................................... 209
  No Sort Field .................................. 209
  Alphabet ...................................... 214
  Letter (Non Locale) ......................... 215
  Letter-Number .................................. 216
  Numerical ................................... 219
  Date-Time ................................... 221
shuffle=\{seed\} .................................. 222
sort-field=\{field\} ................................ 222
missing-sort-fallback=\{field\} ..................... 223
custom-sort-fallbacks=\{\{key=value \list\}\} .......... 224
entry-sort-fallback=\{field\} ...................... 226
abbreviation-sort-fallback=\{field\} ............ 226
symbol-sort-fallback=\{field\} ..................... 228
bibtexentry-sort-fallback=\{field\} ............. 228
field-concat-sep=\{value\} ......................... 229
trim-sort=\{boolean\} ................................ 229
sort-replace=\{list\} ................................ 230
sort-rule=\{value\} ................................ 230
break-at=\{option\} ................................ 233
break-marker=\{marker\} ......................... 235
## Contents

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort-number-pad</td>
<td>\langle number \rangle</td>
<td>235</td>
</tr>
<tr>
<td>sort-pad-plus</td>
<td>\langle marker \rangle</td>
<td>235</td>
</tr>
<tr>
<td>sort-pad-minus</td>
<td>\langle marker \rangle</td>
<td>235</td>
</tr>
<tr>
<td>identical-sort-action</td>
<td>\langle value \rangle</td>
<td>235</td>
</tr>
<tr>
<td>sort-suffix</td>
<td>\langle value \rangle</td>
<td>236</td>
</tr>
<tr>
<td>sort-suffix-marker</td>
<td>\langle value \rangle</td>
<td>241</td>
</tr>
<tr>
<td>strength</td>
<td>\langle value \rangle</td>
<td>241</td>
</tr>
<tr>
<td>decomposition</td>
<td>\langle value \rangle</td>
<td>242</td>
</tr>
<tr>
<td>letter-number-rule</td>
<td>\langle value \rangle</td>
<td>242</td>
</tr>
<tr>
<td>letter-number-punc-rule</td>
<td>\langle value \rangle</td>
<td>243</td>
</tr>
<tr>
<td>numeric-sort-pattern</td>
<td>\langle value \rangle</td>
<td>245</td>
</tr>
<tr>
<td>numeric-locale</td>
<td>\langle value \rangle</td>
<td>245</td>
</tr>
<tr>
<td>date-sort-locale</td>
<td>\langle value \rangle</td>
<td>245</td>
</tr>
<tr>
<td>date-sort-format</td>
<td>\langle value \rangle</td>
<td>245</td>
</tr>
<tr>
<td>group-formation</td>
<td>\langle value \rangle</td>
<td>247</td>
</tr>
<tr>
<td>secondary</td>
<td>\langle value \rangle</td>
<td>248</td>
</tr>
<tr>
<td>secondary-missing-sort-fallback</td>
<td>\langle field \rangle</td>
<td>251</td>
</tr>
<tr>
<td>secondary-trim-sort</td>
<td>\langle boolean \rangle</td>
<td>251</td>
</tr>
<tr>
<td>secondary-sort-replace</td>
<td>\langle list \rangle</td>
<td>251</td>
</tr>
<tr>
<td>secondary-sort-rule</td>
<td>\langle value \rangle</td>
<td>251</td>
</tr>
<tr>
<td>secondary-break-at</td>
<td>\langle value \rangle</td>
<td>251</td>
</tr>
<tr>
<td>secondary-break-marker</td>
<td>\langle marker \rangle</td>
<td>251</td>
</tr>
<tr>
<td>secondary-identical-sort-action</td>
<td>\langle value \rangle</td>
<td>251</td>
</tr>
<tr>
<td>secondary-sort-suffix</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-sort-suffix-marker</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-strength</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-decomposition</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-letter-number-rule</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-letter-number-punc-rule</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-numeric-sort-pattern</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-numeric-locale</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-date-sort-locale</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-date-sort-format</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>secondary-group-formation</td>
<td>\langle value \rangle</td>
<td>252</td>
</tr>
<tr>
<td>dual-prefix</td>
<td>\langle value \rangle</td>
<td>253</td>
</tr>
<tr>
<td>primary-dual-dependency</td>
<td>\langle boolean \rangle</td>
<td>253</td>
</tr>
<tr>
<td>combine-dual-locations</td>
<td>\langle value \rangle</td>
<td>253</td>
</tr>
</tbody>
</table>
Contents

Dual Fields .................................................. 255
dual-type={(value)} ........................................ 255
dual-category={(value)} ................................. 256
dual-counter={(value)} ................................. 256
dual-short-case-change={(value)} ...................... 256
dual-long-case-change={(value)} ...................... 256
dual-field={(value)} ......................................... 257
dual-date-time-field-format={(value)} .............. 257
dual-date-field-format={(value)} ...................... 257
dual-time-field-format={(value)} ...................... 258
dual-date-time-field-locale={(value)} ............... 258
dual-date-field-locale={(value)} ...................... 258
date-time-field-locale={(value)} ...................... 258

Dual Sorting .................................................. 258
dual-sort={(value)} ......................................... 258
dual-sort-field={(field)} ............................... 259
dual-missing-sort-fallback={(field)} .................. 259
dual-trim-sort={(boolean)} ............................. 259
dual-sort-replace={(list)} ............................... 259
dual-sort-rule={(value)} ................................. 259
dual-break-at={(value)} ................................. 259
break-marker={(marker)} ................................. 259
dual-sort-number-pad={(number)} ..................... 259
dual-sort-pad-plus={(marker)} ......................... 259
dual-sort-pad-minus={(marker)} ....................... 259
dual-identical-sort-action={(value)} ............... 259
dual-sort-suffix={(value)} ............................... 259
sort-suffix-marker={(value)} ......................... 260
dual-strength={(value)} ................................. 260
dual-decomposition={(value)} ......................... 260
letter-number-rule={(value)} ......................... 260
letter-number-punc-rule={(value)} ................. 260
numeric-sort-pattern={(value)} ...................... 260
numeric-locale={(value)} ............................... 260
date-sort-locale={(value)} ............................. 260
date-sort-format={(value)} ............................. 260
group-formation={(value)} ............................. 260

Dual Mappings .................................................. 260
dual-entry-map={{{list1}},{{list2}}} ................. 260
dual-abbrv-map={{{list1}},{{list2}}} ................. 262
dual-abbrventry-map={{{list1}},{{list2}}} .......... 262
dual-symbol-map={{{list1}},{{list2}}} ............... 262
dual-indexentry-map={{{list1}},{{list2}}} .......... 262
dual-indexsymbol-map={{{list1}},{{list2}}} ....... 263
dual-indexabbrv-map={{{list1}},{list2}}} .......... 263
Dual Back-Links ............................................. 263
dual-entry-backlink={{boolean}} .................. 263
dual-abbrv-backlink={{boolean}} ................. 264
dual-symbol-backlink={{boolean}} .................. 264
dual-abbrventry-backlink={{boolean}} .......... 264
dual-entryabbr-backlink={{boolean}} ....... 265
dual-indexentry-backlink={{boolean}} ........... 265
dual-indexsymbol-backlink={{boolean}} ........... 265
dual-indexabbrv-backlink={{boolean}} .......... 265
dual-backlink={{boolean}} ................................. 265
5.12 Tertiary Entries ................................. 265
  tertiary-prefix={{value}} ......................... 265
tertiary-type={{value}} ................................. 266
tertiary-category={{value}} ........................... 266
6 Provided Commands 267
6.1 Entry Definitions ................................. 267
  \bibglsnewentry  ....................................... 267
  \bibglsnewsymbol ........................................ 268
  \bibglsnewnumber ....................................... 268
  \bibglsnewindex ........................................ 269
  \bibglsnewindexplural .................................. 269
  \bibglsnewabbreviation .................................. 269
  \bibglsnewacronym ....................................... 270
  \bibglsnewdualentry ..................................... 270
  \bibglsnewdualindexentry .............................. 270
  \bibglsnewdualindexentrysecondary ............... 270
  \bibglsnewdualindexsymbol ............................. 271
  \bibglsnewdualindexsymbolsecondary ............ 271
  \bibglsnewdualindexnumber ............................. 271
  \bibglsnewdualindexnumbersecondary ............ 271
  \bibglsnewdualabbreviation ............................. 272
  \bibglsnewdualabbreviationsecondary .......... 272
  \bibglsnewdualabbreviationentry .................. 273
  \bibglsnewdualabbreviationentrysecondary .. 273
  \bibglsnewdualentryabbreviation .................. 273
  \bibglsnewdualentryabbreviationsecondary .. 274
  \bibglsnewdualsymbol .................................... 274
  \bibglsnewdualnumber ................................... 274
  \bibglsnewdualabbreviation ......................... 275
  \bibglsnewdualacronym ................................... 275
  \bibglsnewtertiaryindexabbreviationentry .... 275
  \bibglsnewtertiaryindexabbreviationentrysecondary .. 276

ix
Contents

\bibglsnewbibtexentry ........................................ 276
\bibglsnewcontributor ........................................ 276
\bibglsnewprogenitor ........................................ 277
\bibglsnewspawnindex ......................................... 277
\bibglsnewspawnedindex ....................................... 277
\bibglsnewspawnindexplural ................................. 277
\bibglsnewspawnedindexplural .............................. 278
\bibglsnewspawnentry ......................................... 278
\bibglsnewspawnedentry ...................................... 278
\bibglsnewspawnabbreviation ............................... 278
\bibglsnewspawnedabbreviation ............................ 279
\bibglsnewspawnacronym ...................................... 279
\bibglsnewspawnedacronym ................................... 279
\bibglsnewspawnsymbol ....................................... 279
\bibglsnewspawnedsymbol ................................... 280
\bibglsnewspawnnumber ....................................... 280
\bibglsnewspawnednumber ................................... 280
\bibglsnewspawnindexsecondary ............................ 280
\bibglsnewspawnedindexsecondary .......................... 281

6.2 Location Lists and Cross-References .................... 281
\bibglsseesep ................................................ 281
\bibglsseealsosep ........................................... 281
\bibglsaliassep .............................................. 282
\bibglsusesee ............................................... 282
\bibglsuseseealso ........................................... 282
\bibglsusealias .............................................. 282
\bibglspaslim ................................................. 282
\bibglspaslimname ............................................ 283
\bibglshrange ................................................ 283
\bibglspinterloper ............................................ 284
\bibglspostlocprefix ......................................... 284
\bibglsprefprefix ............................................. 285
\bibglspagename .............................................. 285
\bibglspagesname ............................................. 286
\bibglspasuffix ............................................... 286
\bibglspasprimary ............................................ 286
\bibglspaslocationgroup ..................................... 286
\bibglspaslocationgroupsep ................................ 287
\bibglssupplemental ........................................... 288
\bibglssupplementalsublist ................................ 288
\bibglssupplementalsep ....................................... 288
Contents

\bibglssupplementalsubsep ........................................... 289
\bibglshrefchar ......................................................... 289
\bibglshrefunicode ..................................................... 289
\bibglshexunicodechar .................................................. 289

6.3 Letter Groups ...................................................... 290
\bibglssetlastgrouptitle ................................................ 292
\bibglssetlettergrouptitle ............................................. 292
\bibglslettergroup ...................................................... 293
\bibglsslettergrouptitle ................................................. 293
\bibglssetothergrouptitle ............................................. 295
\bibglsothergroup ....................................................... 295
\bibglsothergrouptitle .................................................. 296
\bibglssetemptygrouptitle ............................................. 296
\bibglsemptygroup ....................................................... 296
\bibglsemptygrouptitle .................................................. 296
\bibglssetnumbergrouptitle ............................................ 296
\bibglssnumbergroup ..................................................... 297
\bibglssnumbergrouptitle ............................................... 297
\bibglssdatetimegrouptitle ............................................. 297
\bibglsdatetimergroup .................................................. 297
\bibglssdatetimegrouptitle ............................................. 298
\bibglsssetdatetimegrouptitle ......................................... 299
\bibglstimergroup ....................................................... 299
\bibglstimegrouptitle .................................................... 299
\bibglssetunicodedgrouptitle ......................................... 299
\bibglsunicodegroup ..................................................... 299
\bibglssunicodegrouptitle .............................................. 300
\bibglshypergroup ...................................................... 301

6.4 Flattened Entries .................................................. 301
\bibglsfattenedhomograph ............................................. 301
\bibglsfattenedchildpresort .......................................... 303
\bibglsfattenedchildpostsort ......................................... 303

6.5 Other ................................................................. 303
\bibglshyperlink ........................................................ 303
\bibglssetwidest ........................................................ 303
\bibglssetwidestfortype ................................................ 304
\bibglssetwidestfallback ............................................... 304
\bibglssetwidestfortypefallback ...................................... 305
\bibglssetwidesttoplevelfallback .................................... 305
\bibglssetwidesttoplevelfortypefallback ............................ 305
\bibglsscontributorlist ................................................. 305

xi
7 Converting Existing .tex to .bib

7.1 Command Line Arguments

--texenc <encoding> ........................................... 310
--bibenc <encoding> ......................................... 310
--space-sub <replacement> (or -s <replacement>) .......... 310
--ignore-sort ............................................... 311
--no-ignore-sort ........................................... 311
--ignore-type ............................................... 311
--no-ignore-type .......................................... 311
--split-on-type (or -t) .................................... 311
--no-split-on-type ....................................... 311
--ignore-category ......................................... 312
--no-ignore-category ..................................... 312
--split-on-category (or -c) ................................ 312
--no-split-on-category ................................... 312
--ignore-fields <list> (or -f <list>) ..................... 312
--preamble-only (or -p) ................................... 313
--no-preamble-only ....................................... 313
--absorb-see ............................................... 313
--no-absorb-see ......................................... 313
--index-conversion (or -i) ................................ 313
--no-index-conversion .................................... 313
--locale <language tag> .................................... 313
--overwrite ................................................ 314
--no-overwrite ............................................ 314
--silent .................................................... 314
## Contents

<table>
<thead>
<tr>
<th>Option</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>--verbose</td>
<td>314</td>
</tr>
<tr>
<td>--debug</td>
<td>314</td>
</tr>
<tr>
<td>--help (or -h)</td>
<td>314</td>
</tr>
<tr>
<td>--version (or -v)</td>
<td>314</td>
</tr>
</tbody>
</table>

7.2 Recognised Commands
- \texttt{\textbackslash glsexpandfields} .................................................. 315
- \texttt{\textbackslash glsnoexpandfields} ..................................................... 315
- \texttt{\textbackslash glssetexpandfield} ..................................................... 315
- \texttt{\textbackslash glssetnoise\lowercase{xpandfield}} ................................ 316
- \texttt{\textbackslash newglossaryentry} ..................................................... 316
- \texttt{\textbackslash provideglossaryentry} ............................................... 316
- \texttt{\textbackslash longnewglossaryentry} ............................................... 316
- \texttt{\textbackslash longprovideglossaryentry} ........................................... 317
- \texttt{\textbackslash newterm} ....................................................................... 317
- \texttt{\textbackslash newabbreviation} ......................................................... 318
- \texttt{\textbackslash newacronym} .................................................................... 318
- \texttt{\textbackslash glsxtrnewsymbol} ............................................................ 319
- \texttt{\textbackslash glsxtrnewnumber} ............................................................. 319
- \texttt{\textbackslash newdualentry} ................................................................. 320

8 Examples
- no-interpret-preamble.bib ................................................................. 322
- interpret-preamble.bib ............................................................................ 323
- interpret-preamble2.bib ............................................................................ 323
- constants.bib .......................................................................................... 324
- chemicalformula.bib .................................................................................. 327
- bacteria.bib ............................................................................................... 331
- baseunits.bib .............................................................................................. 333
- derivedunits.bib ......................................................................................... 335
- people.bib .................................................................................................. 336
- books.bib .................................................................................................... 342
- films.bib ..................................................................................................... 345
- citations.bib .............................................................................................. 351
- mathgreek.bib ............................................................................................. 352
- bigmathsymbols.bib ..................................................................................... 357
- mathsrelations.bib ....................................................................................... 361
- binaryoperators.bib ..................................................................................... 363
- unaryoperators.bib ....................................................................................... 364
- mathsobjects.bib .......................................................................................... 365
- micsymbols.bib ............................................................................................. 369
- markuplanguages.bib ................................................................................... 373
- usergroups.bib ............................................................................................. 375
- animals.bib .................................................................................................. 380
- minerals.bib .................................................................................................. 382
# List of Tables

2.1 Glossary-Related Commands Implemented by the bib2gls Interpreter .... 22

4.1 Fields Provided by glossaries-extra ........................................ 50
4.2 Fields Provided by bib2gls ..................................................... 51
4.3 Fields Provided by glossaries-prefix ....................................... 51
4.4 Fields Provided by glossaries-accsupp .................................... 51
4.5 Fields Set by bib2gls ........................................................ 52
4.6 Internal Fields Set by glossaries or glossaries-extra or bib2gls .......... 53

5.1 Summary of Available Sort Options: No Sort Field ....................... 210
5.2 Summary of Available Sort Options: Alphabet ............................ 210
5.3 Summary of Available Sort Options: Letter (Non-Locale) ................ 210
5.4 Summary of Available Sort Options: Letter-Number ..................... 210
5.5 Summary of Available Sort Options: Numerical ........................... 211
5.6 Summary of Available Sort Options: Date-Time ........................... 211
List of Figures

5.1 Regular letter comparison vs letter-number comparison .................................. 217

8.1 sample-hierarchical.pdf ......................................................... 389
8.2 sample-nested.pdf ............................................................... 394
8.3 sample-constants.pdf .......................................................... 399
8.4 sample-chemical.pdf ............................................................ 402
8.5 sample-bacteria.pdf ............................................................. 406
8.6 sample-units1.pdf .............................................................. 409
8.7 sample-units2.pdf ............................................................... 412
8.8 sample-units3.pdf .............................................................. 416
8.9 sample-media.pdf ................................................................. 421
8.10 sample-people.pdf ............................................................. 429
8.11 sample-authors.pdf ............................................................. 433
8.12 sample-citations.pdf ............................................................ 437
8.13 sample-msymbols.pdf .......................................................... 440
8.14 sample-maths.pdf .............................................................. 445
8.15 sample-textsymbols.pdf ....................................................... 448
8.16 sample-textsymbols2.pdf ...................................................... 451
8.17 sample-markuplanguages.pdf ................................................ 455
8.18 sample-usergroups.pdf ......................................................... 463
8.19 sample-multi1.pdf (pages 1 to 4) .............................................. 474
8.20 sample-multi1.pdf (pages 5 to 8) .............................................. 475
8.21 sample-multi2.pdf (pages 1 to 4) .............................................. 496
8.22 sample-multi2.pdf (pages 5 to 8) .............................................. 497
8.23 sample-multi2.pdf (pages 9 and 12) ........................................ 498

xvi


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 \TeX{}’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 \write\18 it must be done in the preamble with \immediate. For example:

\immediate\write\18{bib2gls \jobname}

As from version 1.14 of glossaries-extra, this can be done automatically with the automate 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:

```latex
@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:

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

\begin{verbatim}
pdflatex myDoc
bib2gls --group myDoc
pdflatex myDoc
\end{verbatim}

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

\begin{verbatim}
\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: $\gls{v}$. Abbreviation: \gls{html}.
\end{document}
\end{verbatim}
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 glossaries-extra’s point of view, see the supplementary document “glossaries-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 \makeindex methods (described in the main glossaries user manual [14]). As from glossaries-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:

\begin{verbatim}
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
makeglossaries myDoc
pdflatex myDoc
\end{verbatim}

(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 `\glsxtr-setgrouptitle` 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: \textit{type} vs \textit{group} vs \textit{parent}

\begin{verbatim}
\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:

\begin{verbatim}
\printunsrtglossary[style={indexgroup}]
\end{verbatim}

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 \textit{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 \ensuremath{f(x)}},
  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: \textit{type} vs \textit{group} vs \textit{parent}

\begin{verbatim}
@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}
}
\end{verbatim}

This has a custom field \textit{identifier}. This will be ignored by \texttt{bib2gls} unless defined or aliased in the document.

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

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

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`:

```latex
\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 \newglossary to define a new glossary type and some use \newignoredglossary or \newignoredglossary*. Why the starred forms and why define an ignored glossary?

The base glossaries package was originally designed to work with makeindex. Support for xindy was later added, but both require three files per glossary type: the transcript file (created by the indexing application), the file written by \LaTeX (and input by the indexing application) and the file input by \LaTeX (and written by the indexing application). So when a new glossary is defined with \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 \newglossary* is just a shortcut that forms the extensions from the glossary label. For the purposes of bib2gls, this is simpler than the unstarred version since the extensions are now irrelevant as they are only applicable to makeindex and xindy. (Unless, of course, you are using a hybrid method with 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 \newignoredglossary. This only requires an internal control sequence to store the list of entry labels associated with that glossary and the associated internal command that governs the way that commands like \gls are displayed for that glossary type. Since this type of glossary has no associated files, it can’t be used with \printglossary and therefore isn’t included in the list of glossary labels that’s iterated over by commands like \printglossaries. Since there’s no glossary list (and therefore no targets), \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 bib2gls, the indexing is written to the .aux file and so does have an effect.

Although ignored glossaries can’t be used with \printglossary, they can be used with \printunsrtglossary, 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 \printunsrtglossaries, since they’re not included in the list that this command iterates over.

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

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

\footnote{All entries must be assigned to a glossary. If you don’t use the type field the default is used.}
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 .glstex 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 .glstex 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 \(\text{(resource-name)}\)

at this point.

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

Processing resource \(\text{(resource-name)}\)

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

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 .glstex file is written. The transcript will show the message

Selecting entries for resource \langle resource-name \rangle

or (if master)

Processing master \langle resource-name \rangle
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).

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 Indexing

The dual index entries such as \texttt{@dualindexentry} (described in section 4.6) 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}
\begin{document}
\chapter{Example}
\gls{bird}, \gls{html}, $\gls{v}$ and \glspl{goose}.
\printunsrtglossaries
\printindex
\end{document}
\end{verbatim}

If the document is called \texttt{myDoc.tex} then the document build is:

\texttt{pdflatex myDoc}
\texttt{bib2gls myDoc}
\texttt{pdflatex myDoc}
\texttt{makeindex myDoc.idx}
\texttt{pdflatex myDoc}

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

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

1.6 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 \texttt{\glsxtrresourcefile} (and therefore by the shortcut \texttt{\GlsXtrLoadResources}). This extension is fixed and is imposed by both bib2gls and \texttt{\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.

1.7 Localisation

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

The appropriate file is searched for in the following order, where \texttt{\langle locale\rangle} is the operating system’s locale or the value supplied by the --locale switch:

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

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

3. The final attempt is with \texttt{\langle lang\rangle} set to just the two letter ISO language code. For example, bib2gls-sr.xml.
1.8 Conditional Document Build

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.)

Note that if you use the loc-prefix={true} option, the textual labels (“Page” and “Pages” in English) will be taken from the resource file. In the event that the loaded resource file doesn’t match the document language, you will have to manually set the correct translation (in English, this would be loc-prefix={Page,Pages}). The default definition of \bibgls\-passim is also obtained from the resource file.

1.8 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{⟨options⟩}{⟨filename⟩}

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

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

% arara: bib2gls if found("aux", "glsxtr@resource")

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

It might also be possible, although far more complex, to parse the ⟨options⟩ part in each instance of \glsxtr@resource for 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 \glsxtr@resource is found in the .aux file then:
   a) run bib2gls;
   b) run \LaTeX{} (or PDF\LaTeX{} etc).
3. If \@istfilename is found in the .aux file then:
   a) run makeglossaries (or makeglossaries-lite);
   b) run \LaTeX{} (or PDF\LaTeX{} etc).

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

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

Copy the files provided to the following locations:

- \texttt{\langle TEXMF \rangle}/scripts/bib2gls/bib2gls.jar (Java application.)
- \texttt{\langle TEXMF \rangle}/scripts/bib2gls/convertgls2bib.jar (Java application.)
- \texttt{\langle TEXMF \rangle}/scripts/bib2gls/texparslib.jar (Java library.)
- \texttt{\langle TEXMF \rangle}/scripts/bib2gls/resources/bib2gls-en.xml (English resource file.)
- \texttt{\langle TEXMF \rangle}/doc/support/bib2gls/bib2gls.pdf (This document.)
- \texttt{\langle TEXMF \rangle}/doc/support/bib2gls/bib2gls-begin.pdf (Introductory guide.)

If you use the Unix \texttt{man} command, copy the \texttt{bib2gls.1} and \texttt{convertgls2bib.1} files to the appropriate location.

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

\begin{verbatim}
cp bib2gls.sh ~/bin/bib2gls
cp convertgls2bib.sh ~/bin/convertgls2bib
\end{verbatim}

or copy the files to \texttt{\langle TEXMF \rangle}/scripts/bib2gls/ and create a symbolic link to them called just \texttt{bib2gls} and \texttt{convertgls2bib} from somewhere on your path, for example:

\begin{verbatim}
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
\end{verbatim}

The \texttt{texparslib.jar} file isn’t an application but is a library used by both \texttt{bib2gls.jar} and \texttt{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 .\texttt{bat} file that works in a similar way to the bash scripts. To do this, create a file called \texttt{bib2gls.bat} that contains the following:

\begin{verbatim}
@ECHO OFF
FOR /F "tokens=" %%I IN (kpsewhich --prognam=bib2gls --format=texmfscripts bib2gls.jar') DO SET JARPATH=%%I
java -Djava.locale.providers=CLDR,JRE -jar "%JARPATH%"
\end{verbatim}
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:

\begin{verbatim}
bib2gls --version
convertgls2bib --version
\end{verbatim}

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

The \texttt{bib2gls} application requires the \TeX\ Parser Library \texttt{texparserlib.jar} which is used to parse the .aux and .bib files.

With the \texttt{--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 \texttt{sort} option is not \texttt{unsrt} or \texttt{none} or \texttt{use}).

The other cases that the interpreter is used for are:

- when \texttt{set-widest} is used to determine the width of the \texttt{name} field;
- if \texttt{labelify} or \texttt{labelify-list} are set the identified field values are first interpreted (if they contain \ \{ \} - or $) before being converted to labels;
- if \texttt{interpret-label-fields={true}} is set and the \texttt{parent}, \texttt{category}, \texttt{type}, \texttt{group}, \texttt{seealso} or \texttt{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 \texttt{inputenc} to prevent expansion when they are written to the .aux file. (In some options, such as \texttt{sort-rule}, you can use \texttt{\glshex\{hex\}} syntax to specify a UTF-8 character.)

The \texttt{--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 \texttt{see} field doesn’t use the interpreter with \texttt{interpret-label-fields={true}} as it may legitimately contain $\LaTeX$ code in the optional tag part (such as \texttt{\seealso{name} or \also{name}}).

The parser has a different concept of expansion to \TeX\ and will expand some things that aren’t expanded by $\LaTeX$ (such as \texttt{\MakeUppercase} and \texttt{\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 \texttt{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 \texttt{\foo} that has been defined as:

\footnote{https://github.com/nlct/texparser}

\footnote{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 \texttt{\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 \textup{\LaTeX} document as well (unless the document has a different definition that doesn’t have this unbounded recursion).

The \texttt{texparserlib.jar} library is not a \textup{\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.

\texttt{bib2gls} can detect from the log file a small number of packages that the parser recognises. Note that in some cases there’s only very limited support. For example, \texttt{siunitx’s \si} command is recognised but other commands from that package aren’t. See \texttt{--list-known-packages} (page 34) for further details.

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

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 \texttt{--custom-packages} switch. Note that commands that reference an entry, such as \texttt{\glsentryname}, aren’t guaranteed to work across resource sets and will only be able to look up field values that are known to \texttt{bib2gls}. (For example, the \texttt{name} field for abbreviations is typically set by \texttt{bib2gls}.)

If a command isn’t recognised, you can provide it in the \texttt{@preamble} and use \texttt{\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 \texttt{\Harbour}, \texttt{\Battlefield} and \texttt{\Stadium}, then you can provide versions of these commands just for \texttt{bib2gls}’s use:³

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

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

\textup{\LaTeX} syntax can be quite complicated and, in some cases, far too complicated for simple regular expressions. The \textup{\LaTeX} parser library performs better than a simple pattern match, and that’s the purpose of \texttt{texparserlib.jar} and why it’s used by \texttt{bib2gls} (and by \texttt{convert-gls2bib}). When the \texttt{--debug} mode is on, any warnings or errors triggered by the interpreter will be written to the transcript prefixed with \texttt{texparserlib:} (the results of the conversions will be included in the transcript as informational messages prefixed with \texttt{texparserlib:} even with \texttt{--no-debug}).

³These commands won’t work with PDF\textup{\LaTeX}, as the \texttt{\char} values are too large, but they’re fine for \texttt{bib2gls}.
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>\bibglsampersand</td>
<td>\bibglscircum</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>\glsentryuseri</td>
<td>\Glsentryuseri</td>
</tr>
<tr>
<td>\glsentryuserii</td>
<td>\Glsentryuserii</td>
<td>\glsentryuseri</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>\gllopenbrace</td>
</tr>
<tr>
<td>\glspercentchar</td>
<td>\glstildechar</td>
<td>\GlsXtrEnableInitialTagging</td>
</tr>
<tr>
<td>\glsxthiername</td>
<td>\Glsxthiername</td>
<td>\GlsXtrhiernamename</td>
</tr>
<tr>
<td>\GLSXthiername</td>
<td>\glsxthiernamesep</td>
<td>\glsxthiernamesep</td>
</tr>
<tr>
<td>\glsxtprovidecommand</td>
<td>\glsxtrusefield</td>
<td>\Glsxtrusefield</td>
</tr>
<tr>
<td>\Glsxtrusefield</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For example, suppose the `.bib` file includes:

```latex
@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 ($\sqrt{-1}$)}
}
```

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

None of these entries have a `sort` field so the `name` is used. If the entry type had been `@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 bib2gls will use texparserlib.jar to interpret this code. The library has several different ways of writing the processed code. For simplicity, 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 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`:

\providecommand{\mtx}{\[1\]{\textbf{#1}}}  \
\providecommand{\set}{\[1\]{\mathcal{#1}}}  \
\providecommand{\card}{\[1\]{|\set{#1}|}}  \
\providecommand{\imaginary}{i}

(unless `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:

{}\$\mtx{M}\$

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

texparserlib: {}\$\mtx{M}\$ -> M

So the sort value for this entry is set to “$M$”. The font change (caused by math-mode and \textbf) 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:

{}\$\vec{v}\$

The transcript shows:

texparserlib: {}\$\vec{v}\$ -> $\vec{v}$

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:

{}\$\set{S}\$

The transcript shows:
texparserlib: \{\texttt{\textbackslash set}\{S\}\texttt{\textbackslash \}} \rightarrow S

So the \texttt{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:

\{\texttt{\textbackslash card}\{S\}\texttt{\textbackslash \}}

The transcript shows:

texparserlib: \{\texttt{\textbackslash card}\{S\}\texttt{\textbackslash \}} \rightarrow |S|

So the \texttt{sort} value for this entry is set to “|S|” (the | characters from the definition of \texttt{card} provided in \texttt{@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 \texttt{interpret-preamble=\{false\}} had been used, \texttt{\textbackslash card} wouldn’t be recognised and would be discarded leaving just “S” as the sort value.

(Note that if \texttt{\textbackslash 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:

\{\texttt{\textbackslash imaginary}\texttt{\textbackslash \}}

The transcript shows:

texparserlib: \{\texttt{\textbackslash imaginary}\texttt{\textbackslash \}} \rightarrow i

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

This means that in the case of the default \texttt{\textbackslash sort-field=\{sort\}} with \texttt{\textbackslash 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 \texttt{bib2gls} with \texttt{--verbose} the decimal character codes will be included in the transcript. For this example:

\begin{verbatim}
i -> 'i' [105]
card -> '|S|' [124 83 124]
M -> 'M' [77]
S -> 'S' [83]
v -> 'v' [118 8407]
\end{verbatim}

The \texttt{--group} option (in addition to \texttt{--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.

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

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

\footnote{For 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 (\(|S|\)), \text{angstrom} (Å), i (i), M (M), S (S), and v (\vec{v}). The --group switch shows that the \text{angstrom} entry (Å) has been placed in the “A” letter group.

However, if I change the locale to `sort={sv}`, the \text{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 \text{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 Java Runtime Environment (JRE). For example:

```
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 \text{bib2gls}, for example,

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

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

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

or in Windows:

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

---

5 A better method is to use \text{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.

--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.
3 Command Line Options

--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.

--silent

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.

--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 sort={doc} is used and the document doesn’t have any language support. Note that sort={locale} uses the Java Virtual Machine’s (jvm) default 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.

--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.

--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

\GlsXtrLoadResources[src={test-entries}]


3 Command Line Options

\texttt{pdflatex -output-directory tmp mydoc}

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

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

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

If I run \texttt{bib2gls} from the same directory as \texttt{mydoc.tex} using

\texttt{bib2gls tmp/mydoc}

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

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

\texttt{bib2gls -d tmp mydoc}

\texttt{--interpret}

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

\texttt{--no-interpret}

Switch off the interpreter mode. See chapter 2 for more details about the interpreter.

\texttt{--no-break-space}

The interpreter treats a tilde character ~ as a non-breakable space (default). Similarly \texttt{\nobreakspace} produces a non-breakable space character (0x00A0).

\texttt{--break-space}

The interpreter treats a tilde character ~ as a normal space. Similarly \texttt{\nobreakspace} just produces a space.
3 Command Line 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 92).

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

--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.
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 Command Line Options

--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 save-index-counter.)

--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.

--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:

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

the record is written as:

\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 \the\Hentrycounter, which is set to the hyper target command \the\H⟨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⟨counter⟩ can’t be formed from ⟨prefix⟩\the⟨counter⟩, 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 ⟨href⟩ (to link to the nearest anchor) or ⟨hcounter⟩ to link to the place where the indexing counter was incremented.

The nameref record will be written to the location list using:

\glsxtrdisplaylocnameref{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨title⟩}{⟨href⟩}{⟨hcounter⟩}{⟨file⟩}

The ⟨file⟩ part will be empty for normal internal locations, and will be set to the corresponding file name for supplemental locations.

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

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

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

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

\texttt{--force-cross-resource-refs (or -x)}

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

\texttt{--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).

\texttt{--support-unicode-script}

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

\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 \texttt{<sup>} and \texttt{<sub>} markup, which is then stripped by bib2gls. For example,

\textsuperscript{(2,3)}

will be converted to

<sup>(2,3)</sup>
(since there’s no superscript comma). The markup is stripped leaving just \((2,3)\).

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.

\textbf{--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.

\textbf{--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, amssymb, bpchem, fontenc, fontspec, fourier, hyperref, lipsym, 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 fontspec is to simply determine whether or not UTF-8 characters are allowed in labels (for labelify and labelify-list).

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.
3 Command Line Options

--packages ⟨list⟩ (or -p ⟨list⟩)
Instruct the interpreter to assume the packages listed in ⟨list⟩ 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.

--custom-packages ⟨list⟩
Instruct the interpreter to parse the package files identified in ⟨list⟩. 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 ⟨list⟩ (or -k ⟨list⟩)
This option is cumulative. When the document .log file is parsed for known packages, bib2gls will skip the check for any listed in ⟨list⟩. 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 ⟨list⟩.

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

```latex
@abbreviation{shtml,
  short={shtml},
  long={\glspss{ssi} enabled \glsp{html}}
}
```

The default mfirstuc protection will automatically insert an empty group before \glsp{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). 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 --no-mfirstuc-protection. The default value is all.

--no-mfirstuc-protection

Switches off the mfirstuc protection mechanism described above.

--mfirstuc-math-protection

This works in the same way as --mfirstuc-protection but guards against fields starting with inline maths ($…$). For example, if the 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 --mfirstuc-math-protection set, 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.

--no-mfirstuc-math-protection

Switches off the above.

--nested-link-check ⟨list⟩|none

By default, 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 ⟨value⟩ where ⟨value⟩ may be none (don’t parse any of the fields) or a comma-separated list of fields to be checked.
3 Command Line Options

--no-nested-link-check

Equivalent to --nested-link-check none.

--shortcuts ⟨value⟩

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 ⟨value⟩ (where ⟨value⟩ may take any of the allowed values for the shortcuts package option), but in general there is little need to use this switch.

--map-format ⟨map:value list⟩ (or -m ⟨map:value list⟩)

This sets up the rule of precedence for partial location matches (see section 5.7). 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,

```
bib2gls --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:

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

and bib2gls is invoked with

```
bib2gls --map-format "emph:hyperbf,hypersf:hyperit" mydoc
```

or

```
bib2gls --map-format emph:hyperbf --map-format hypersf:hyperit mydoc
```

then bib2gls 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 bib2gls consults the mappings provided by --map-format.
   - The hypersf format (from the new record) is mapped to hyperit, so bib2gls checks if the existing record has this format. In this case it doesn’t (the format is emph). So bib2gls moves on to the next test:
   - 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

   --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 primary reference.)

--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 letter 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-unsrtglossary 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 `nogroupskip =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.

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 `\bibglsnumbergroup`.

**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`.

39
custom group The group label is explicitly set either by aliasing a field (with \texttt{field-aliases}) or by using the \texttt{group=\{\langle label\rangle\}} resource option. You will need to use \texttt{\glsxtrset-group} in the document to provide an associated title if the \texttt{\langle label\rangle} isn’t the same as the title. Remember that the label can’t contain any active characters, so you can’t use non-ASCII characters in \texttt{\langle label\rangle} with inputenc (but you can use non-ASCII alphanumerics with fontspec).

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:

\begin{verbatim}
<entry key="grouptitle.case.ij">IJ</entry>
\end{verbatim}

If there isn’t a \texttt{grouptitle.case.\langle lc\rangle} key (where \texttt{\langle lc\rangle} 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 \texttt{Character.isAlphabetic(int) method}} then it will be a letter group otherwise it’s a non-letter group. The case-insensitive ordering (such as \texttt{sort={letter-nocase}}) will convert the letter group character to upper case. The case-sensitive ordering (such as \texttt{sort={letter-case}}) won’t change the case.

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

\begin{verbatim}
@gls@hypergroup{\langle type\rangle}{\langle group id\rangle}
\end{verbatim}

The third \texttt{pdflatex} reads this information and is then able to create the navigation links.
With \texttt{bib2gls}, if the \texttt{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 .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:
\begin{verbatim}
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record,abbreviations,style={indexhypergroup}]{glossaries-extra}

\GlsXtrLoadResources[src={entries},% data in entries.bib
  \texttt{type}={main}% put these entries in the 'main' glossary
]
\GlsXtrLoadResources[src={abbrvs},% data in abbrvs.bib
  \texttt{type}={abbreviations}% put entries in the 'abbreviations' glossary
]

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

\texttt{pdflatex myDoc}
\texttt{bibtex --group myDoc}
\texttt{pdflatex myDoc}

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

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

For example:
\begin{verbatim}
\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:

\texttt{pdflatex myDoc}
\texttt{bibtex --group myDoc}
\texttt{pdflatex myDoc}
\texttt{pdflatex myDoc}
\end{verbatim}
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).

--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 fontsfp 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 JVM’s 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 (.g1g) for the line that starts

  \TeX\ character encoding:

  This should be followed by the encoding used by bib2gls when creating the .g1stex 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 Command Line 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 programmatically (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`).

--trim-fields

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

```latex
@endentry{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 `.glstex` 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 `\ `, `\space` or `\~`.

--trim-only-fields ⟨list⟩

Only trim leading and trailing spaces from the fields identified in the comma-separated ⟨list⟩. 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.

For example, to only trim the description field:

```
bib2gls --trim-only-fields description myDoc
```

**--trim-except-fields** *(list)*

Trim all leading and trailing spaces from fields except those identified in the comma-separated *(list)*. 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.

**--provide-glossaries**

This setting will make *bib2gls* add the line

```
\provideignoredglossary*{(type)}
```

to the .glistex 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.
3 Command Line Options

--record-count (or -c)

Switch on record counting. This will ensure that when each entry is written to the .glistex 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. 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 \glsxtrenablerecordcount 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}]
\glsxtrenablerecordcount
\GlsXtrSetRecordCountAttribute{abbreviation}{1}

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

--no-record-count

Switch off record counting. (Default.)
---record-count-unit (or -n)

Automatically implements --record-count and additionally sets the `recordcount.{counter}.<location>` fields. These fields can then be used with the `\rgls`-like commands.

---no-record-count-unit

Switches off unit record counting. (Default.) Note that you need --no-record-count to completely switch off record counting.
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 XƎLₐₜₑₓ or Luaₐₜₑₓ, 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

The original *.bib* file format as defined by BnTTeX doesn’t have a designated comment character, but instead treats anything outside of `@⟨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. BnTTeX’s `@comment` is also supported by bib2gls for general comments, but not for the encoding. For example, JabRef uses `@Comment` for metadata.

```latex
@Comment{jabref-meta: databaseType:bib2gls;}
```

### 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⟩{⟨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.

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.
4.3 Fields

Internal fields that may be assigned within the document (the \LaTeX assignment code having been written by \texttt{bib2gls} in the \texttt{.glistex} file) are listed in Table 4.5. These typically shouldn’t be used in the \texttt{.bib} file. Some of these fields can be set for a particular document using a resource option, such as \texttt{type} or \texttt{group}. With \texttt{--warn-non-bib-fields} set, \texttt{bib2gls} will check for internal fields that can cause interference with its normal operations and will warn if any are found in the \texttt{.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 \texttt{bib2gls}. In most cases these fields don’t have a designated key and are only intended for internal use by \texttt{bib2gls} or by the glossaries or glossaries-extra package. Note that the value of the \texttt{sort} field written to the \texttt{.bib} file doesn’t always exactly match the sort value used by \texttt{bib2gls} (which is stored in \texttt{bib2gls@sort}). Any special characters found in the sort value are always substituted before writing the \texttt{.bib} file to avoid syntax errors.

Any unrecognised fields will be ignored by \texttt{bib2gls}. This is more convenient than using \texttt{\input} or \texttt{\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 \texttt{see}, \texttt{seealso} or \texttt{alias} fields or by using commands like \texttt{\gls} or \texttt{\glsxtrp} in any of the recognised fields. These will automatically be selected if the \texttt{selection} setting includes dependencies, but you may need to rebuild the document to ensure the location lists are correct. Use of the \texttt{\glssee} command will create an ignored record and the \texttt{see} field will be set to the relevant information. If an entry has the \texttt{see} field already set, any instance of \texttt{\glssee} in the document for that entry will be appended to the \texttt{see} field (provided you have at least v1.14 of glossaries-extra). In general, it’s best just to use the \texttt{see} field and not use \texttt{\glssee}.

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

You can identify an arbitrary field as containing a list of dependent entry labels with \texttt{dependency-fields}. This instructs \texttt{bib2gls} to parse the listed fields for dependencies in a similar manner to the \texttt{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 \texttt{see} or \texttt{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>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>
<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>
</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>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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bibtexcontributor</td>
<td>An internal list field provided when a</td>
</tr>
<tr>
<td></td>
<td>@contributor entry is automatically created by</td>
</tr>
<tr>
<td></td>
<td>@bibtextentry.</td>
</tr>
<tr>
<td>bibtexentry</td>
<td>An internal list field created by @bibtextentry.</td>
</tr>
<tr>
<td>bibtextentry⟨entry-type⟩</td>
<td>An internal list field created by @bibtextentry.</td>
</tr>
<tr>
<td>bibtexttype</td>
<td>Used by bib2gls as a substitution for LaTeX’s</td>
</tr>
<tr>
<td></td>
<td>type field when parsing @bibtextentry. Needs to</td>
</tr>
<tr>
<td></td>
<td>be defined or aliased to make it available in the</td>
</tr>
<tr>
<td></td>
<td>document.</td>
</tr>
<tr>
<td>childcount</td>
<td>Stores the number of children this entry has</td>
</tr>
<tr>
<td></td>
<td>had selected.</td>
</tr>
<tr>
<td>childlist</td>
<td>A list of labels (in etoolbox’s internal list</td>
</tr>
<tr>
<td></td>
<td>format) of the children this entry has selected.</td>
</tr>
<tr>
<td>counter</td>
<td>The default counter used for indexing (assigned by</td>
</tr>
<tr>
<td></td>
<td>the counter option).</td>
</tr>
<tr>
<td>dual</td>
<td>Created by dual-field if set with no value, this</td>
</tr>
<tr>
<td></td>
<td>field is used to store the dual label.</td>
</tr>
<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</td>
</tr>
<tr>
<td></td>
<td>assigned by the group option). See section 1.2.</td>
</tr>
</tbody>
</table>
Fields Sometimes Set by bib2gls in the .glistex File (Continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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>desc</td>
<td>Corresponds to description key.</td>
</tr>
<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>
</tbody>
</table>

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

Internal Fields Set by `glossaries` or `glossaries-extra` or `bib2gls`

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>descplural</td>
<td>Corresponds to <code>descriptionplural</code> key.</td>
</tr>
<tr>
<td>firstpl</td>
<td>Corresponds to <code>firstplural</code> 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 (<code>makeindex</code> or <code>xindy</code>).</td>
</tr>
<tr>
<td>level</td>
<td>Hierarchical level.</td>
</tr>
<tr>
<td>longpl</td>
<td>Corresponds to <code>longplural</code> key.</td>
</tr>
<tr>
<td>prevcount</td>
<td>Used with entry counting to store the total from the previous run.</td>
</tr>
<tr>
<td>prevcount(&lt;\text{value})&gt;</td>
<td>Used with unit entry counting (<code>glossaries-extra</code>).</td>
</tr>
<tr>
<td>prevunitmax</td>
<td>Used with unit entry counting (<code>glossaries-extra</code>).</td>
</tr>
<tr>
<td>prevunittotal</td>
<td>Used with unit entry counting (<code>glossaries-extra</code>).</td>
</tr>
<tr>
<td>shortpl</td>
<td>Corresponds to <code>shortplural</code> 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 (<code>glossaries-extra</code>).</td>
</tr>
<tr>
<td>useri</td>
<td>Corresponds to <code>user1</code> key.</td>
</tr>
<tr>
<td>userii</td>
<td>Corresponds to <code>user2</code> key.</td>
</tr>
<tr>
<td>useriii</td>
<td>Corresponds to <code>user3</code> key.</td>
</tr>
<tr>
<td>useriv</td>
<td>Corresponds to <code>user4</code> key.</td>
</tr>
<tr>
<td>userv</td>
<td>Corresponds to <code>user5</code> key.</td>
</tr>
<tr>
<td>uservi</td>
<td>Corresponds to <code>user6</code> key.</td>
</tr>
</tbody>
</table>
4.4 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",
  long=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:

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

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

Alternatively you can use \glsxtrprovidecommand which behaves the same as \providecommand within the document but behaves like \renewcommand within bib2gls, which al-
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 \TeX{} parser library used by \texttt{bib2gls} will parse the contents of \texttt{@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:

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

\begin{verbatim}
@entry{S,
  name={}$\set{S}$$,
  text={}$\set{S}$,
  description={a set}
}
\end{verbatim}

\begin{verbatim}
@entry{card,
  name={}$\card{S}$$,
  text={}$\card{S}$,
  description={the cardinality of \gls{S}}
}
\end{verbatim}

Neither entry has the \texttt{sort} field, so \texttt{bib2gls} has to fall back on the \texttt{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 \texttt{bib2gls} to deduce that the \texttt{sort} value of the \texttt{S} entry is just \texttt{S} and the \texttt{sort} value of the \texttt{card} entry is \texttt{|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 \texttt{bib2gls} won't be able to correctly interpret the \texttt{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 .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 @preamble definitions) and in the second example the @preamble definitions will take precedence. For example, the document may define \card as:

\newcommand{\card}{\vert\set{#1}\vert}

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

The .bib file can now look like this:

@preamble{\providecommand{\set}{\mathcal{#1}} \providecommand{\card}{\vert\set{#1}\vert}"

@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⟩ using the formatting command provided in the user1 field with:

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

(which internally uses \glslink) or

\glsxxtreentryfmt{⟨label⟩}{⟨text⟩}

which just applies the appropriate formatting command to ⟨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 \langle label \rangle hasn’t been defined, then \glsxtrfmt just does \langle text \rangle (followed by \langle insert \rangle 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 \langle options \rangle are as for \glslink but \glslink will actually be using:

\glslink[(def-options),\langle options\rangle]{{\langle label\rangle}\{\langle csname\rangle{{\langle text\rangle}\langle insert\rangle}}

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

This means that the document doesn’t need to actually provide \set or \card but can instead use, for example,

\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 .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 .bib that defines entries with a term, a description and a symbol:

@preamble{"\providecommand{\setfmt}[1]{\mathcal{#1}}
\providecommand{\cardfmt}[1]{|\setfmt{#1}|}"
}@entry{set,
  name={set},
  symbol={\setfmt{S}},
  user1={\setfmt},

4.4 Standard Entry Types

description={collection of values}

@entry{cardinality,
  name={cardinality},
  symbol={\cardfmt{S}},
  user1={cardfmt},
  description={the number of elements in the \gls{set} $\glssymbol{set}$}
}

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:

\newcommand{\gset}[2][]{\glsxtrfmt[#1]{set}{#2}}
\newcommand{\gcard}[2][]{\glsxtrfmt[#1]{cardinality}{#2}}

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:

@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 \gls{set} $\glssymbol{set}$}
}

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

\newcommand{\gset}[1][]{\glsxentryfmt{set}{#1}}
\newcommand{\gcard}[2][]{\glsxentryfmt[#1]{cardinality}{#2}}

This allows for code such as:

[ \gcard{\gset{A} \cap \gset{B}} ]
4.5 Single Entry Types

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:

\newcommand{\gset}[1]{\glsxtrentryfmt{set}{#1}}
\newcommand{\gcard}[1]{\glsxtrentryfmt{cardinality}{#1}}

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

4.5 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:

@entry{matrix,
   name={matrix},
   plural={matrices},
   description={rectangular array of values}
}

is analogous to:

\newglossaryentry{matrix}% label
{% fields
   name={matrix},
   plural={matrices},
   description={rectangular array of values}
}

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 4.6 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}},}
4.5 Single Entry Types

```latex
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}
}

@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.

Terms defined using @entry will be written to the output (.gls.tex) 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:

```latex
@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 sort 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.

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

@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}
}

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

This stores the approximate value in the 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 category={number} option makes it easy to adjust the glossary format to include the user1 field:

\glsdefpostdesc{number}{% 
  \ifglsfield{useri}{\glscurrententrylabel}{}
  { (approximate value: \glscurrentfieldvalue)}%
{}%
}

@index

The @index entry type is designed for entries that don’t have a description. Only the label is required. If name is omitted, it’s assumed to be the same as the label, even if parent is present. (Note this is different to the fallback behaviour of @entry, which fetches the name from the parent entry.) If the name contains any characters that can’t be used in the label,

62
you must use the `name` field. If the `sort` field is missing the default is obtained from the `name`. Note that the `@index` entry type is *not* governed by `entry-sort-fallback` (but it is governed by `custom-sort-fallbacks`). This allows `@index` and `@entry` to have different fallbacks if the `sort` field is missing.

Example:

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

Terms that are defined using `@index` will be written to the output file using the command `\bibglsnewindex`.

### @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`.) All fields are optional. For example:

```latex
@indexplural{goose, plural = {geese}}
@indexplural{duck}
@indexplural{chateau, text = {ch\^ateau}, plural = {ch\^ateaux}}
```

This is equivalent to:

```latex
@indexplural{goose, name = {geese}, text = {goose}, plural = {geese}}
@indexplural{duck, name = {ducks}, text = {duck}, plural = {ducks}}
```
4.5 Single Entry Types

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.

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:

@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):

\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:
4.5 Single Entry Types

\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:

\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 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:

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

then the document may include:

\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:

\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.
4.6 Dual Entry Types

@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 .glistex file using the command \bibglsnewcontributor. There are no required fields. As with @index, if the name field is missing, the fallback value is the entry’s label. When this entry type is automatically created by @bibtexentry, the name is set to

\bibglscontributor{{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 bib-texentry@⟨entry-type⟩ internal fields can be set for the @contributor entry. For example:

@contributor{KnuthDonaldE,
    name={\bibglscontributor{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.

4.6 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 (also referred to as the secondary entry, but is not related to the secondary option). The naming scheme is @dual⟨entry-type⟩ where both the primary
and dual are considered to have the same type of entry (such as \texttt{@dualsymbol} where both the primary and dual are functionally like \texttt{@symbol} or \texttt{@dual(primary)(dual)} where the primary is functionally like \texttt{@(primary)} and the dual is functionally like \texttt{@(dual)}.

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:

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

is like:

\begin{verbatim}
@abbreviation{svm,
  short = {SVM},
  long = {support vector machine},
}
@entry{dual.svm,
  text = {SVM},
  name = {support vector machine},
  description = {statistical pattern recognition technique}
}
\end{verbatim}

and is analogous to:

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

but both entries are considered dependent on each other. This means that if you only reference the primary entry (using \texttt{\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. Any fields that aren’t listed in the mappings are simply copied, except for the \texttt{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 \texttt{<prefix>(id)} where \texttt{<prefix>} is set by the dual-prefix option and \texttt{<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.
4.6 Dual Entry Types

Take care if you have a mixture of entry types (such as `@dualindexentry`, `@dualindexsymbol` 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 4.6 (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 `@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 .glstex file so that the internal list of labels is in the appropriate order.

For example, suppose the file entries.bib contains:

```latex
@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:

```latex
\documentclass{article}
```

68
\usepackage[record,abbreviations]{glossaries-extra}
\GlsXtrLoadResources[selection=all,src=entries]
\begin{document}
\printunsrtglossaries
\end{document}

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 .glstex file is essentially (but not exactly):

% 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}

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

% primary of @dualindexabbreviation{html,…}:
\newabbreviation{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={}}
4.6 Dual Entry Types

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

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

Since the document uses the \texttt{abbreviations} package option, \texttt{\newabbreviation} automatically assigns the abbreviation to the \texttt{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 \texttt{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 \texttt{short} for the abbreviations and \texttt{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 \texttt{\gls{css}} and \texttt{\gls{ssi}} for the primary abbreviations created with \texttt{@dualabbreviationentry} and \texttt{\gls{dual.html}} and \texttt{\gls{dual.xml}} for the dual abbreviations created with \texttt{@dualindexabbreviation}. Also the \texttt{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}]

70
then bib2gls first orders the primaries:

- **aardvark** (name: aardvark),
- **css** (short: CSS),
- **html** (name: HTML),
- **mouse** (name: mouse),
- **ssi** (short: SSI),
- **xml** (name: XML),
- **zebra** (name: zebra)

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

```latex
\begin{itemize}
\item \texttt{\dualabbreviationentry{css,...}:} \\
\newglossaryentry{dual.css}{name={cascading stylesheets},text={CSS},
\description={a language that describes the style of an \texttt{html} document}}
\item \texttt{\dualindexabbreviation{html,...}:} \\
\newabbreviation{dual.html}{HTML}{hypertext markup language}
\item \texttt{\dualabbreviationentry{ssi,...}:} \\
\newglossaryentry{dual.ssi}{name={server-side includes},text={SSI},
\description={directives placed in \texttt{html} pages evaluated by the server}}
\item \texttt{\dualindexabbreviation{xml,...}:} \\
\newabbreviation{dual.xml}{XML}{extensible markup language}
\end{itemize}
```

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),
4.6 Dual Entry Types

- xml (XML),
- zebra (zebra),
- dual.css (cascading stylesheets),
- 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 ↦ description
- plural ↦ descriptionplural
- description ↦ name
- descriptionplural ↦ plural

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

- prefix ↦ dualprefix
- prefixplural ↦ dualprefixplural
- prefixfirst ↦ dualprefixfirst
- prefixfirstplural ↦ dualprefixfirstplural
- dualprefix ↦ prefix
4.6 Dual Entry Types

- 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'
]
4.6 Dual Entry Types

@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 $\mapsto$ name

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

This doesn’t actually perform any swapping of fields, but it provides the field used for backlinks (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:

@index{array}

@entry{dual.array,
  name={array},
  description={ordered list of values}
}

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:
4.6 Dual Entry Types

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},
  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}.
\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 \glsxtraddlabeledprefix and reference entries with \dgls, \dGls etc instead of \gls, \Gls etc.

@dualindexabbreviation

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

- name $\mapsto$ name

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

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.

For example:

@dualindexabbreviation{html,
    short = {HTML},
    long = {hypertext markup language} 
}
4.6 Dual Entry Types

is like:

@index{html, name={HTML}}

@abbreviation{dual.html,
    short = {HTML},
    long = {hypertext markup language}
}

The primary term is defined using \bibglssnewdualindexabbreviation, 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 \bibglss-useabbrvfont is used, otherwise \bibglssuselongfont 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 \bibglssnewdualindexabbreviation-secondary.

@dualindexsymbol

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

- symbol $\mapsto$ name
- name $\mapsto$ symbol
- symbolplural $\mapsto$ plural
- plural $\mapsto$ symbolplural

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

78
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 `\bibglsnewdualindexsymbol`, which by default sets the `category` to `index`, and the dual entries are defined using `\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}}}
\end{verbatim}

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

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

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

\begin{verbatim}
\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}% }
\end{verbatim}

79
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
4.6 Dual Entry Types

and it also switches off the hyperlinks (with `hyper={false}`). These custom commands are defined using:

\begin{verbatim}
\glsxtrnewglslike[⟨options⟩]{⟨prefix⟩}{⟨gls-like cs⟩}{⟨glspl-like cs⟩}{⟨Gls-like cs⟩}{⟨Glsxpl-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:

```
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
```

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

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

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

- \texttt{prefix} \mapsto \texttt{dualprefix}
- \texttt{prefixplural} \mapsto \texttt{dualprefixplural}
- \texttt{prefixfirst} \mapsto \texttt{dualprefixfirst}
- \texttt{prefixfirstplural} \mapsto \texttt{dualprefixfirstplural}
- \texttt{dualprefix} \mapsto \texttt{prefix}
- \texttt{dualprefixplural} \mapsto \texttt{prefixplural}
- \texttt{dualprefixfirst} \mapsto \texttt{prefixfirst}
- \texttt{dualprefixfirstplural} \mapsto \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 \dual⟨primary⟩⟨dual⟩.)

The required fields are: \texttt{short}, \texttt{long} and \texttt{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 \texttt{description}) and the secondary entry is a regular entry with the \texttt{name} copied from the \texttt{long} field. The fallback for the \texttt{sort} is given by \texttt{abbreviation-sort-fallback}, which defaults to the \texttt{short} field.

For example:

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

is rather like doing:
4.6 Dual Entry Types

@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:

\bibglssnewdualabbreviationentry

(which uses \newabbreviation) for the primary entries and with :

\bibglssnewdualabbreviationentrysecondary

(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.

@dualsymbol

This is like @dualentry but the default mappings are:

- name → symbol
- plural → symbolplural
- symbol → name
- symbolplural → plural

If the prefix fields are defined, then the default mappings additionally include:
4.6 Dual Entry Types

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

The \texttt{name} and \texttt{symbol} fields are required. For example:

\begin{verbatim}
@dualsymbol{pi,
    name={pi},
    symbol={\ensuremath{\pi}},
    description={the ratio of the length of the circumference of a circle to its diameter}
}
\end{verbatim}

Entries are defined using \texttt{\bibglsnewdualsymbol}, which by default sets the \texttt{category} to \texttt{symbol}.

@\textbf{dualnumber}

This is almost identical to @\texttt{dualsymbol} but entries are defined using \texttt{\bibglsnewdualnumber}, which by default sets the \texttt{category} to \texttt{number}.

The above example could be defined as a number since \(\pi\) is a constant:

\begin{verbatim}
@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}
}
\end{verbatim}

This has stored the approximate value in the \texttt{user1} field. The post-description hook could then be adapted to show this.

\begin{verbatim}
\glsdefpostdesc{number}{%\
    \ifglsishasfield{useri}{\glscurrententrylabel}{(approximate value: \glscurrentfieldvalue)}%\
    {}}%\}
\end{verbatim}
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` $\mapsto$ `dualshort`
- `shortplural` $\mapsto$ `dualshortplural`
- `long` $\mapsto$ `duallong`
- `longplural` $\mapsto$ `duallongplural`
- `dualshort` $\mapsto$ `short`
- `dualshortplural` $\mapsto$ `shortplural`
- `duallong` $\mapsto$ `long`
- `duallongplural` $\mapsto$ `longplural`

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: 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 .glistex file with
\glsxtrprovidestoragekey\{\langle key\rangle\}\{}\{}

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 .glistex 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\}.

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 \glsdefpostdesc).

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}
}\}

If the abbreviation is in the file called entries-dual-abbrv.bib, then here’s an example document:
\documentclass\{article\}
\usepackage\{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\}
4.6 Dual Entry Types

\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 primary 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  
\end{document}  
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  
\textit{ribonucleic acid (RNA, Ribonukleinsäure)}  
and the first use of the dual entry is displayed as:
4.6 Dual Entry Types

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:
4.6 Dual Entry Types

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},
  {duallong,duallongplural,dualshort,dualshortplural,
   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:

dual-abbrv-map={%
  {long,longplural,short,shortplural,dualshort,dualshortplural,
   duallong,duallongplural,description,user1},
  {duallong,duallongplural,dualshort,dualshortplural,
   short,shortplural,long,longplural,user1,description}
%
}%

Then the back-link field will switch to duallong. The post-description hook can be modified
to allow for this:

\glsdefpostdesc{abbreviation}{%\ifglshasfield{duallong}{\glscurrententrylabel}%
  \space(\glscurrentfieldvalue)%
%}{%}

An alternative is to use the long–short–user–desc style without the post-description hook:
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 and second of these are the primary and secondary. The third of these is the tertiary which is effectively an appendage of the 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.

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:

@tertiaryindexabbreviationentry{html,  
  short = {HTML},  
  long = {hypertext markup language},  
  description = {a markup language for creating web pages}  
}

is analogous to:
The last two are actually defined using one command:

\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:

\providecommand{\bibglsnewtertiaryindexabbreviationentrysecondary}[8]{%
 \newabbreviation[#3]{#1}{#6}{#7}%
 \longnewglossaryentry*{#2}{name={\protect\bibglsuselongfont{#7}{\glscategory{#1}}},#4}%
 {#8}%
}

which defines the secondary as an abbreviation using \newabbreviation and the tertiary as a regular entry using \longnewglossaryentry. This means that the tertiary entry is always defined immediately after the corresponding secondary entry. The primary may be defined earlier or later in the file depending on the way the entries are sorted and on the dual-sort setting.

4.8 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.
4.8 Multi-Entry Types

@bibtexentry

The @bibtexentry type will typically need to be aliased as it’s designed for converting \TeX entries into \bib2gls entries. For example, to make \bib2gls treat @article and @book as though they were both @bibtexentry:

```
entry-type-aliases={
  article=bibtexentry,
  book=bibtexentry
}
```

For convenience, \glossaries-\extra-\bib2gls v1.29+ provides `\GlsXtrBibTeXEntryAliases` which covers all the standard \TeX 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`. If you want to access any of the \TeX fields, you will need to alias or define them. For example:

```
field-aliases={
  title=name
}
```

Since \TeX’s `type` field conflicts with \bib2gls’s `type` field, when \bib2gls parses @bibtexentry if will convert `type` to `bibtextype`, so you must use `bibtextype` as the identifier when aliasing.

Alternatively, you can use `\GlsXtrProvideBibTeXFields` which uses `\glsaddstorage-key` to provide all the standard \TeX fields. (Remember that new fields must be defined before the first resource set.)

The @bibtexentry essentially creates an @index form of entry, but it additionally defines a @contributor entry for each listed author or editor and updates the dependency lists: each @contributor is added to the main @bibtexentry’s list of dependencies (so if the @bibtexentry has a record then all its satellite @contributors are selected with the default `selection={recorded and deps}`), and each @contributor is treated as having a cross-reference to the main @bibtexentry (so if a @contributor has a record then all the linked @bibtexentry terms will be selected if `selection={recorded and deps and see}`). You can instruct \bib2gls to treat \citation as an ignored record using `--cite-as-record`.

Each contributor is effectively defined as:

```
@contributor{{⟨label⟩},
  name={\bibglscontributor{⟨forenames⟩}{⟨von⟩}{⟨surname⟩}{⟨suffix⟩}}
}
```
The label is obtained by converting the name to a label, using the same function as labelify (which means it’s governed by labelify-replace).

The author and editor fields are always checked, even if those fields aren’t recognised by 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 @contributor entry has already been defined, it won’t be redefined, but will have the new @bibtexentry added to its internal bibtexentry field.

The main @bibtexentry is defined using \bibglsnewbibtexentry and is followed by:

\glsxtrfieldlistadd{⟨id⟩}{bibtexcontributor}{⟨contributor-id⟩}

where ⟨id⟩ is the label identifying the main @bibtexentry and ⟨contributor-id⟩ is the label identifying the contributor, for each contributor that has been selected.

Each contributor is defined using \bibglsnewcontributor. The definition is followed by:

\glsxtrfieldlistadd{⟨contributor-id⟩}{bibtexentry}{⟨id⟩}
\glsxtrfieldlistadd{⟨contributor-id⟩}{bibtexentry@⟨entry-type⟩}{⟨id⟩}

for each selected @bibtexentry associated with that contributor. The second line provides the internal list field bibtexentry@⟨entry-type⟩, where ⟨entry-type⟩ is the original entry type (before it was aliased to @bibtexentry and converted to lower case). For example article or book.

You can iterate over these internal list fields using \glsxtrfielddolistloop or \glsxtrfieldforlistloop. For example:

\newcommand{\contributorhandler}[1]{\par\glsentryname{#1}}
\newcommand{\glsxtrpostdesccontributor}{\%
\glsxtrifhasfield{bibtexentry}{\glscurrententrylabel}\
{\%
\glsxtrfielddolistloop
{\glscurrententrylabel}{bibtexentry}%
{\contributorhandler}%
}%
{\par No titles.}%}

(where 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}
4.8 Multi-Entry Types

\newcommand{\bibglsnewbibtlexentry}[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} ]

\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}{\glsreffieldname{\glscurrententrylabel}{bibtexentry}}{\% 
    \glsxtrfieldforlistloop 
    {\glscurrententrylabel}{bibtexentry}{\par\glsentryname{#1} (1)}% 
    \par No titles.}% 
\begin{document} 

94
4.8 Multi-Entry Types

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}]
{%
\renewcommand{\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

@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.
4.8 Multi-Entry Types

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:

```plaintext
@progenitor{(id),
  adoptparents = {(parent-1 id),...,{parent-N id)},
  ⟨field-name-1⟩ = {(text)},
...
  ⟨field-name-n⟩ = {(text)}
}
```

is essentially like:

```plaintext
@index{(id),
  progeny = {(parent-1 id) . {id} ,..., {parent-N id) . {id}},
  ⟨field-name-1⟩ = {(text)},
...
  ⟨field-name-n⟩ = {(text)}
}
```

```plaintext
@index{(parent-1 id) . {id},
  progenitor = {{id}},
  parent = {{parent-1 id}},
  ⟨field-name-1⟩ = {(text)},
...
  ⟨field-name-n⟩ = {(text)}
}
```

```plaintext
@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 adoptparents) 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.
4.8 Multi-Entry Types

In addition to the main \langle id \rangle entry, the above also creates the spawned progeny entries \langle parent-1 id \rangle . \langle id \rangle, \ldots, \langle parent-N id \rangle . \langle id \rangle that are dependent on the main \langle id \rangle entry.

The spawned entries have the parent field set to the corresponding label obtained from the adoptparents 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 .gllstex file with \bibglsnew-spawnedindex

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 \langle parent-id \rangle . \langle id \rangle) 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},
    adoptparents={markup}
}

@progenitor{css,
    name={CSS},
    adoptparents={stylesheet,webdesign}
}

@progenitor{html,
    name={HTML},
    adoptparents={markup,webdesign}
}

@progenitor{xsl,
    name={XSL},
    adoptparents={stylesheet}
}

and if the document contains:
\documentclass{article}
4.8 Multi-Entry Types

\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

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} entry. These special \texttt{\@spawned\{type\}} entry types aren’t intended for use in the \texttt{.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 \texttt{.glistex} file with \texttt{\bibglsnewspawnindex} and the spawned entries are defined with \texttt{\bibglsnewspawnedindex}.
4.8 Multi-Entry Types

@spawnindexplural
As @progenitor, except that it creates @indexplural terms instead of @index. As with @indexplural, if the name field isn’t set, it’s assigned to the same value as the plural field (or the fallback for the plural, if not defined).

The main entries are defined in the .glstex file with \bibglsnewspawnindexplural and the spawned entries are defined with \bibglsnewspawnedindexplural.

@spawnentry
As @progenitor, except that it creates @entry terms instead of @index. As with @entry, the description field is required and either name or parent.

The main entries are defined in the .glstex file with \bibglsnewspawnentry and the spawned entries are defined with \bibglsnewspawnedentry.

@spawnabbreviation
As @progenitor, except that it creates @abbreviation terms instead of @index. As with @abbreviation, the short and long fields are required.

The main entries are defined in the .glstex file with \bibglsnewspawnabbreviation and the spawned entries are defined with \bibglsnewspawnedabbreviation.

@spawnacronym
As @progenitor, except that it creates @acronym terms instead of @index. As with @acronym, the short and long fields are required.

The main entries are defined in the .glstex file with \bibglsnewspawnacronym and the spawned entries are defined with \bibglsnewspawnedacronym.

@spawnsymbol
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 .glstex file with \bibglsnewspawnsymbol 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 .glstex file with \bibglsnewspawnnumber and the spawned entries are defined with \bibglsnewspawnednumber.
4.8 Multi-Entry Types

@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 .glstex file with \bibglsnewspawndualindexentry and \bibglsnewspawndualindexentrysecondary. The spawned progeny are defined with \bibglsnewspawndindex.
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 \BibTeX to write it to any external files (unless you use the hybrid `record={alsoindex}`, 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 \BibTeX 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 \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 \texttt{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}}
\GlsXtrLoadResources[ 
   type={symbols},
   match={entrytype=symbol}]
\GlsXtrLoadResources[ 
   type={abbreviations},
   match={entrytype=abbreviation}]
\end{verbatim}

This acts like:

\begin{verbatim}
\GlsXtrLoadResources[ 
   selection={all},src={entries},
   type={symbols},
   match={entrytype=symbol}]
\GlsXtrLoadResources[ 
   selection={all},src={entries},
   type={abbreviations},
   match={entrytype=abbreviation}]
\end{verbatim}

If you have multiple .bib files you can either select them all using \texttt{src={\langle bib list\rangle}} in a single \Glsxtrresourcefile call, if they all require the same settings, or you can load them separately with different settings applied.

For example, if the files entries-terms.bib and entries-symbols.bib have the same settings:

\begin{verbatim}
\GlsXtrLoadResources[src={entries-terms,entries-symbols}]\end{verbatim}

Alternatively, if they have different settings:

\begin{verbatim}
\GlsXtrLoadResources[src={entries-terms},type={main}]
\GlsXtrLoadResources[src={entries-symbols},sort={use},type={symbols}]\end{verbatim}
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:

```latex
@index{duck}
@index{zebra}
@index{aardvark}
```

and `file2.bib` contains:

```latex
@index{caterpillar}
@index{bee}
@index{wombat}
```

then

```latex
\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

```latex
\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 `\printunsrtglossary` 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:

```latex
@preamble{"\providecommand{\strong}[1]{\textbf{\color{red}#1}}
\providecommand{\parenswap}[2]{#1 (#2 (#1))}"
```

This provides two commands:

```latex
\strong{⟨text⟩}
```

(which sets the font weight and colour) and

```latex
\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={\strong{\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\strong{\parenswap{stuff}{example}}```

to:

```latex\textbf{\color{red}example (stuff)}```

It discards font changes, so \textbf is ignored, but it doesn’t recognise \color 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\strong{\parenswap{stuff}{example}}```

no expansion occurs (since \strong and \parenswap 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{\strong}[1]{\textbf{\color{red}#1}}"}```

and defs-interpret.bib:

```latex\@preamble{"providecommand{\parenswap}[2]{#2 (#1)}"}````
Now the first one can be loaded with \texttt{interpret-preamble={false}}:

\GlsXtrLoadResources[\texttt{src={defs-noindent},interpret-preamble={false}}]

This creates a \texttt{.glstex} file that provides \texttt{strong} but doesn’t define any entries. The other file \texttt{defs-interpret.bib} can then be loaded with the default \texttt{interpret-preamble={true}}:

\GlsXtrLoadResources[\texttt{src={defs-interpret,entries}}]

The provided commands are remembered by the interpreter, so you can also do:

\GlsXtrLoadResources[\texttt{src={defs-interpret}}]
\GlsXtrLoadResources[\texttt{src={entries}}]

The \texttt{contents} of \texttt{@preamble} are only written to the associated \texttt{.glstex} file, but the definitions contained within the \texttt{@preamble} are retained by the interpreter for subsequent resource sets.

### 5.1 General Options

\textbf{charset=⟨encoding-name⟩}

If the character encoding hasn’t been supplied in the \texttt{.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 \texttt{.bib} file where it can also be read by a \texttt{.bib} managing systems, such as JabRef.

See \texttt{--tex-encoding} for the encoding used to write the \texttt{.glstex} file.

\textbf{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 \texttt{.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).

\textbf{write-preamble=⟨boolean⟩}

This is a boolean option that determines whether or not the preamble should be written to the \texttt{.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 \LaTeX.
**set-widest=⟨boolean⟩**

The alttree glossary style needs to know the widest **name** (for each level, if hierarchical). This can be set using \glssetwidest provided by the glossary-tree package (or similar commands like \glsupdatewidest provided by glossaries-extra-stylemods), but this requires knowing which name is the widest. Alternatively, one of the iterative commands such as \glsFindWidestTopLevelName can be used, which slows the document build as it has to iterate over all defined entries.

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 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 \glsextrSetWidest, 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:

```
name={some \Huge huge text}
```

and another entry has the **name** defined as:

```
name={some \small small text}
```

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;
5.1 General Options

- otherwise \bibglsetwidest 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 \bibglsetwidestfortypefallback is used,
    - otherwise \bibglsetwidesttoplevelfortypefallback is used;
  - otherwise \bibglsetwidestfortype is used.

This leaves \TeX to compute the width according to the document fonts. If \texttt{bib2gls} can’t correctly determine the widest entry then you will need to use one of the commands provided by \texttt{glossary-tree}, \texttt{glossary-longextra} or \texttt{glossaries-extra-stylemods} to set it.

In general, if you have more than one glossary it’s best to set the type using options like \texttt{type} and \texttt{dual-type} if you use \texttt{set-widest}.

\texttt{entry-type-aliases=\{key=value list\}}

In the .\texttt{bib} file, the data is identified by @\texttt{(entry-type)}, such as @\texttt{abbreviation}. It may be that you want to replace all instances of @\texttt{(entry-type)} with a different type of entry. For example, suppose my .\texttt{bib} file contains abbreviations defined in the form:

\begin{verbatim}
@abbreviation{html,
  short = {html},
  long = {hypertext markup language},
  description = {a markup language for creating web pages}
}
\end{verbatim}

but suppose in one of my documents I actually want all these abbreviations defined with @\texttt{dualabbreviationentry} instead of @\texttt{abbreviation}. Instead of editing the .\texttt{bib} file I can just supply a mapping:

\begin{verbatim}
\GlsXtrLoadResources[
  src={entries},% data in entries.bib
  entry-type-aliases={abbreviation=dualabbreviationentry}
]
\end{verbatim}

This makes all instances of @\texttt{abbreviation} behave as @\texttt{dualabbreviationentry}. You can have more than one mapping. For example:

\begin{verbatim}
\GlsXtrLoadResources[
  src={entries},% data in entries.bib
  entry-type-aliases={
    % @abbreviation -> @dualabbreviationentry:
    abbreviation=dualabbreviationentry,
    % @entry -> @index:
    entry=index
  }
]
\end{verbatim}
5.1 General Options

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:

```
@foo{html,
    name = {HTML},
    short = {HTML},
    long = {hypertext markup language},
    description = {hypertext markup language}
}
```

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}
}
```
5.1 General Options

**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.

**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 produce 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. 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.

109
5.1 General Options

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` 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:

```
\newignore{glossary}{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
5.2 Selection Options

glossary. The simplest way to do this is with the starred form of \printunsrtglossary. For example, if \texttt{copy-action-group-field=\{dupgroup\}} is added to the options for the second resource set:

\begin{verbatim}
\printunsrtglossary*\{type={copies},title={Duplicates},style={indexgroup}\}
\renewcommand{\glsxtrgroupfield}{dupgroup}
\end{verbatim}

This just does:

\begin{verbatim}
\begingroup
\renewcommand{\glsxtrgroupfield}{dupgroup}\
\printunsrtglossary[type={copies},title={Duplicates},style={indexgroup}]
\endgroup
\end{verbatim}

5.2 Selection Options

\texttt{src=\{list\}}

This identifies the .\texttt{bib} files containing the entry definitions. The value should be a comma-separated list of the required .\texttt{bib} files. These may either be in the current working directory or in the directory given by the \texttt{--dir} switch or on \LaTeX{}’s path (in which case \texttt{kpsewhich} will be used to find them). The .\texttt{bib} extension may be omitted. Remember that if \texttt{\{list\}} contains multiple files it must be grouped to protect the comma from the \texttt{\{options\}} list.

For example:

\begin{verbatim}
\GlsXtrLoadResources[src={entries-terms,entries-symbols}]
\end{verbatim}

indicates that \texttt{bib2gls} must read the files entries-terms.bib and entries-symbols.bib and create the file given by \texttt{\jobname.glstex} on the first instance or \texttt{\jobname-\{n\}.glstex} on subsequent use.

With \texttt{\glsxtrresourcefile\{\{options\}\\{\{filename\}\}}, if the \texttt{src} option is omitted, the .\texttt{bib} file is assumed to be \texttt{\{filename\}}.\texttt{bib}. For example:

\begin{verbatim}
\glsxtrresourcefile{entries-symbols}
\end{verbatim}

indicates that \texttt{bib2gls} needs to read the file entries-symbols.bib, which contains the entry data, and create the file entries-symbols.glstex. If the .\texttt{bib} file is different or if you have multiple .\texttt{bib} files, you need to use the \texttt{src} option.

\texttt{\GlsXtrLoadResources} uses \texttt{\jobname} as the argument of \texttt{\glsxtrresourcefile} on the first instance, so:

\begin{verbatim}
\GlsXtrLoadResources[]
\end{verbatim}

will assume \texttt{src=\{\jobname\}}. Remember that subsequent uses of \texttt{\GlsXtrLoadResources} append a suffix, so in general it’s best to always supply \texttt{src}. 

111
5.2 Selection Options

If you have non-ASCII characters in the .bib filename but aren’t using \[Xe\TeX\] or \[Lua\TeX\], then you will need to use `detokenize` to prevent expansion when the information is written to the .aux file. Similarly for any special characters that need protecting (although it’s better not to use special characters in filenames). For example:

```latex
\documentclass{article}
\usepackage[T2A]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[russian]{babel}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[
  src={\detokenize{кириллица}},% data in кириллица.bib
  selection={all}
]
\begin{document}
\printunsrtglossary
\end{document}
```

`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:

- **recorded** and **deps**: add all recorded entries and their dependencies (default).
- **recorded** and **deps** and **see**: as above but will also add unrecorded entries whose `see`, `seealso` or `alias` field refers to a recorded entry.
- **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.
- **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
\texttt{\textbackslash gls} in the field values that are parsed by \texttt{bib2gls}. With this setting, parents will be omitted unless they’ve been referenced in the document through commands like \texttt{\textbackslash gls}.

- **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.

- **deps but not recorded**: this first selects entries as though \texttt{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 \texttt{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 \texttt{sample-nested.tex} example document.

- **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 \texttt{entries.bib} contains:

```latex
@index{run}
@index{sprint,see={run}}
@index{dash,see={sprint}}
```

If the document only references the “run” entry (for example, using \texttt{\textbackslash gls\{run\}}) then:

- **If 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 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:

```latex
@index{run}
@index{sprint,see={run}}
@index{dash,see={run}}
```
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:

```bib
@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:

```bib
@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},
```
Now, whichever one is indexed in the document, the other two will automatically be selected.

\texttt{\textbf{match}=\langle \texttt{key=value list} \rangle}

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 \langle \texttt{key=value list} \rangle should be a \langle \texttt{key}=\langle \texttt{regexp} \rangle list, where \langle \texttt{key} \rangle 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 \texttt{type} field identifying the glossary label). If you’ve used \texttt{entry-type-aliases}, this refers to the target entry type not the original entry type specified in the .bib file.

The \langle \texttt{regexp} \rangle part should be a regular expression conforming to Java’s Pattern class [5]. The pattern is anchored (oo.* matches oops but not loops) and \langle \texttt{regexp} \rangle can’t be empty. Remember that \LaTeX{} 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 \texttt{\textbackslash string\.} not \texttt{\textbackslash}. (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 \langle \texttt{key=value list} \rangle will be ignored.

If a field is listed multiple times, the pattern for that field is concatenated using:

\texttt{(?:\langle pattern-1 \rangle)|(?:\langle pattern-2 \rangle)}

where \langle \texttt{pattern-1} \rangle is the current pattern for that field and \langle \texttt{pattern-2} \rangle is the new pattern. This means it performs a logical OR. For the non-duplicate fields the logical operator is given by \texttt{match-op}. For example:

\begin{verbatim}
match-op=\{and\},
match=\{
    category=animals,
    topic=biology,
    category=vegetables
\}
\end{verbatim}

This will keep all the selected entries that satisfy:

- \texttt{category} matches \texttt{\langle?:animals\rangle|\langle?:vegetables\rangle}
  (the \texttt{category} is either \texttt{animals} or \texttt{vegetables})

\textbf{AND}

- \texttt{topic} (custom key provided by user) is \texttt{biology}.

115
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 \texttt{match=\{\}}. That is, no filtering will be applied. In the above
example, the custom topic key must be provided before the first \texttt{\GlsXtrLoadResources}
with \texttt{\glsaddkey} or \texttt{\glsaddstoragekey}.

\textbf{match-op=\langle value \rangle}

If the value of \texttt{match} contains more than one \texttt{\langle key\rangle=\langle pattern \rangle} element, the \texttt{match-op}
determines whether to apply a logical AND or a logical OR. The \texttt{\langle value \rangle} may be either and or or. The default is \texttt{match-op=\{and\}}.

\textbf{not-match=\langle key=value list \rangle}

If \texttt{match=\{\langle key=value list \rangle\}} would cause an entry to be selected then \texttt{not-match=\{\langle key=value list \rangle\}} 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 \texttt{match} and
\texttt{not-match} in the same resource set, the last one listed takes precedence.

\textbf{match-action=\langle value \rangle}

The default behaviour with \texttt{match} or \texttt{not-match} is to filter the selection. This may be
changed to append to the selection instead. The \texttt{\langle value \rangle} may be one of:

- \texttt{filter}: (default) filter selection;
- \texttt{add}: append any matches (with \texttt{match}) or non-matches (with \texttt{not-match}) to the se-
lection. This setting can’t be used with \texttt{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:

\begin{verbatim}
\GlsXtrLoadResources[
  src={entries},% data in entries.bib
  match-action={add},
  match={category=important}
]
\end{verbatim}

\textbf{limit=\langle number \rangle}

If \texttt{\langle number \rangle} is greater than 0 then this will truncate the list of selected entries after sorting
to \texttt{\langle number \rangle} (if the list size is greater than that value). The transcript will show the message:

\texttt{Truncating according to limit=\langle number \rangle}
5.3 Hierarchical Options

When used with `shuffle`, this provides a means of randomly selecting at most \( \langle \text{number} \rangle \) entries. The default setting is `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 `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 `.glistex` file, so this will increase the total number of entries.

### 5.3 Hierarchical Options

**save-child-count=\langle \text{boolean} \rangle**

This is a boolean option. The default setting is `save-child-count={false}`. If `save-child-count={true}`, each entry will be assigned a field called `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 `childlist` field for entries that have children selected. This field is in etoolbox’s internal list format and can be iterated over using `\glsxtrfieldforlistloop`.

The assignment is done using `\GlsXtrSetField` so there’s no associated key. You can test if the field is set and non-zero using:

\[
\text{\GlsXtrIfHasNonZeroChildCount}{\langle \text{entry label} \rangle}{\langle \text{true} \rangle}{\langle \text{false} \rangle}
\]

which is provided with `glossaries-extra-bib2gls v1.31+`. Within `\langle \text{true} \rangle`, you can access the actual value with `\glscurrentfieldvalue`. If `save-child-count={false}`, this command will do `\langle \text{false} \rangle` as the `childcount` field won’t be set.

For example, suppose `entries.bib` contains:

```biblatex
@index{birds}
@index{duck,parent={birds}}
@index{goose,plural={geese},parent={birds}}
@index{swan,parent={birds}}

@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:

```latex
\documentclass{article}
\usepackage[record,style={indexgroup}]{glossaries-extra}
```

117
5.3 Hierarchical Options

\GlsXtrLoadResources[src={entries},save-child-count]
\begin{document}
\gls{duck} and \gls{goose}.
\gls{quartz}, \gls{corundum}, \gls{amethyst}.
\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}{% 
   \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.
5.3 Hierarchical Options

**save-sibling-count=⟨boolean⟩**

This is a boolean option. The default setting is `save-sibling-count={false}`. This is like `save-child-count` but saves the sibling count in `siblingcount` and the sibling list in `siblinglist`. As with the child list, 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 `siblinglist` over accessing the parent's `childlist` is that the entry itself is excluded from the list.

**flatten=⟨boolean⟩**

This is a boolean option. The default value is `flatten={false}`. If `flatten={true}`, the sorting will ignore hierarchy and the `parent` field will be omitted when writing the definitions to the `.glistex` 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 `ignore-fields={parent}` which will remove the dependency (unless a dependency is established through another field).

**flatten-lonely=⟨value⟩**

This may take one of three values: `false` (default), `presort` and `postsort`. The value must be supplied.

Unlike the `flatten` option, which completely removes the hierarchy, the `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 `flatten-lonely-rule` setting. The conditions for moving a child up one hierarchical level are as follows:

- 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 `see` or `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
- `flatten-lonely-rule` isn’t set to `no discard`.

The value of `flatten-lonely` determines whether the adjustment should be made before sorting (`presort`) or after sorting (`postsort`). To disable this function use `flatten-lonely={false}`.

For example, suppose the file `entries.bib` contains:
Although the duck entry has siblings in the entries.bib file, none of them have been recorded (indexed) 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:
Hierarchical Options

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 “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
The name field of the duck entry has been set to:
\begin{verbatim}
name={\bibglsflattenedchildpostsort{birds}{duck}}
\end{verbatim}
the text field has been set to:
\begin{verbatim}
text={duck}
\end{verbatim}
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:
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:

```latex
name=
{\%
\bibglsflattenedchildpostsort
{\%
  % name from former parent
  \bibglsflattenedchildpostsort{item}{subitem}%
}\%
{subsubitem}% original name
}
```

The first argument of \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:

```latex
\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
```
5.3 Hierarchical Options

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
This method uses a different format for the modified `name` field. For example, the duck entry now has:

\[\text{name} = \{\text{\textbackslash bibglsflattenedchildpresort\{duck\}\{birds\}}\}\]

The default definition of `\bibglsflattenedchildpresort` simply does the first argument and ignores the second. The sorting is then performed, but the interpreter recognizes 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:

\[\text{@preamble\{"\textbackslash providecommand\{\textbackslash bibglsflattenedchildpresort\} [2]\{#1 \ (#2)\}"\}}\]

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:

\[\text{\textbackslash bibglsflattenedhomograph\{(name)\}\{(parent label)\}}\]

instead of the corresponding `\bibglsflattenedchild...sort`. This defaults to just `(name)`.
5.3 Hierarchical Options

**flatten-lonely-rule=**\langle value\rangle

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:

- **only unrecorded parents**  Only the sub-entries that have a parent without a location list will be altered. The parent entry will be removed from the selection. This value is the default setting.

- **discard unrecorded** This setting will adjust all sub-entries that have no siblings 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.

- **no discard** This setting will adjust all sub-entries that don’t have siblings 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.

**strip-missing-parents=**\langle boolean\rangle

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 `.glistex` 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 ‘\langle parent id\rangle’ not found for entry ‘\langle child-id\rangle’

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=**\langle value\rangle

As an alternative to `strip-missing-parents`, as from version 1.4 you can now use `missing-parents=\langle value\rangle` where `\langle value\rangle` may be one of:

- **strip**: this is equivalent to `strip-missing-parents={true}`;
5.3 Hierarchical Options

- **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:

```latex
@entry{ubik,
    name={Ubik},
    description={novel by Philip K. Dick},
    identifier={book},
    author={\sortmediacreator{Philip K.}{Dick}},
    year={1969}
}
```

then the field alias:

```latex
field-aliases={author=parent}
```

will treat:

```latex
author={\sortmediacreator{Philip K.}{Dick}},
```

as though it had been defined as:

```latex
parent={\sortmediacreator{Philip K.}{Dick}},
```

This can be converted into a label with the options:

```latex
labelify={parent},
labelify-replace={
    {[ \string\.]}}
```

If the interpreter has been provided with the definition:

```latex
\providecommand*{\sortmediacreator}[2]{#2 \ #1}
```

then the `parent` field for the `ubik` 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 `ubik` 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:

```latex
@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: 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}.

5.4 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:
5.4 Master Documents

\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, \makeglossaries and \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 bib2gls. First a .bib file needs to be created:

\@entry{sample,\nname={sample},\ndescription={an example}}

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:

\documentclass{article}
\usepackage{colorlinks}{hyperref}
5.4 Master Documents

\usepackage[record]{glossaries-extra}
\newignoredglossary*{external}
\glssetcategoryattribute{external}{targeturl}{mybook.pdf}
\glssetcategoryattribute{external}{targetname}{\glolinkprefix\glslabel}
\GlsXtrLoadResources[
  src={myentries},
  sort={none},
  type={external},
  category={external}
]
\begin{document}
\gls{sample}.
\end{document}

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.

**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}.

The use of master isn’t always suitable. In particular if any of the terms cross-reference each other, such as through the see or seealso 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 position.

When bib2gls detects the master option, it won’t search for entries in any .bib files (for that particular resource set) but will create a .glstex file that inputs the master document’s
5.4 Master Documents

...glstex files, but it will additionally temporarily adjust the internal commands used to define entries so that the prefix given by label-prefix, the glossary type and the category type are all automatically inserted. If the type or category options haven’t been used, the corresponding value will default to master. The targeturl and targetname category attributes will automatically be set, and the glossary type will be provided using \provideignore-glossary*{(type)} (even if --no-provide-glossaries is set).

The above myarticle.tex can be changed to:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[label-prefix={book.},master={mybook}]
\begin{document}
\gls{book.sample}.
\end{document}

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 label-prefix={term.} then you also need this prefix when you reference the entries in the dependent document in addition to the label-prefix for the dependent document. Suppose mybook.tex loads the resources using:

\GlsXtrLoadResources[src={myentries},label-prefix={term.}]

and myarticle.tex loads the resources using:

\GlsXtrLoadResources[label-prefix={book.},master={mybook}]

Then the entries referenced in myarticle.tex need to use the prefix book.term. as in:

This is a \gls{book.term.sample} term.

Remember that the category labels will need adjusting to reflect the change in category label in the dependent document.

For example, if mybook.tex included:

\setabbreviationstyle{long-short-sc}

then myarticle.tex will need:

\setabbreviationstyle[master]{long-short-sc}

(change master to ⟨value⟩ if you have used category=⟨⟨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.
5.5 Field and Label Options

**master-resources\{list\}**

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 \(\langle\text{list}\rangle\) is a comma-separated list of names, where each name corresponds to the final argument of \texttt{\glsxtrresource-file}. 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 \(\langle\text{list}\rangle\)) The file \texttt{\jobname.glstex} is considered the primary resource file and the files \texttt{\jobname-\langle n\rangle.glstex} (starting with \(\langle n\rangle\) 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}

5.5 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**

**interpret-label-fields\{boolean\}**

This is a boolean option that determines whether or not the fields that may only contain labels should have their values interpreted (parent, category, type, group, seealso and alias). Although this option interprets commands within those fields, it doesn’t strip any characters that can’t be used within a label. The see field isn’t included as it may optionally start with \![\langle tag\rangle] where \(\langle tag\rangle\) may legitimately contain \LaTeX code that shouldn’t be interpreted.

The default setting is \texttt{interpret-label-fields={false}}. Note that if this setting is on, cross-resource references aren’t permitted. This setting has no effect if the interpreter has been disabled.

Related settings are \texttt{labelify} and \texttt{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 \texttt{labelify} and \texttt{labelify-list} options are performed before \texttt{interpret-label-fields}.
5.5 Field and Label Options

\texttt{labelify=\langle list \rangle}

This option should take a comma-separated list of recognised field names as the value. (If a field is present in both \texttt{labelify} and \texttt{labelify-list}, then \texttt{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 \texttt{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 \texttt{labelify-list} instead. If you want to strip any of the allowed punctuation, use \texttt{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 \texttt{bib2gls} hasn’t detected \texttt{fontspec} in the document’s transcript file, the value is then decomposed and all non-ASCII characters are removed.

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

\begin{verbatim}
@index{sample,  
  name={\AA ngstr"om, \O stergaard, d'Arcy, and Fotheringay-Smythe}  
}
\end{verbatim}

Then:

\begin{verbatim}
\GlsXtrLoadResources[  
  src={entries},% data in entries.bib  
  labelify={name}  
]
\end{verbatim}

will convert the \texttt{name} field into:

\texttt{Angstrom stergaard d'Arcy and Fotheringay-Smythe}

if the document hasn’t used \texttt{fontspec} otherwise it will be:
5.5 Field and Label Options

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 non-fontspec 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 ATEX or LuaL ATEX.

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:

Ångstrom, stergaard, d'Arcy, and Fotheringay-Smythe

or:

Ångström, Østergaard, d'Arcy, and Fotheringay-Smythe

depending on whether or not fontspec was detected.

labelify-replace=⟨list⟩

This option takes a comma-separated list as a value with each element in the list in the form \{(regex)\}{⟨replacement⟩} where \(regex\) is a regular expression (that conforms to Java’s Pattern class [5]) and \(replacement\) 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 \GlsXtrLoadResources is expanded when written to the .aux file so take care to protect any special characters. For example, to match a literal full stop use \string\ . rather than just \ . (backslash dot).

In the \(replacement\) part, you can use \glscapturedgroup\(n\) to reference a captured sub-sequence. For example:
5.5 Field and Label Options

\labelify-replace=\{\{([A-Z])\string\.\}\{\glscapturedgroup1}\}\}

This removes any full stop that follows any of the characters A,...,Z. Alternatively, you can just use \string\$ instead of \glscapturedgroup. If you want a literal dollar character, you need to use \glshex24 (or \string\u24). This isn't recommended for labels (since special characters are automatically stripped), but sort-replace follows the same rules as labelify-replace, and it may be needed for that.

Both labelify and labelify-list use the labelify-replace setting to perform substitutions. For example, to replace the sub-string " and " (including spaces) with a comma:

\GlsXtrLoadResources[
\src={entries},\% data in entries.bib\n\replicate-fields={name={user1}},\n\labelify-replace=\{ and \}{,},\n\labelify-list={user1}\n]

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 fontspec was detected.

Note that this produces the same result regardless of whether or not the Oxford comma is present as ,\,\text{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:

\GlsXtrLoadResources[
\src={entries},\% data in entries.bib\n\replicate-fields={name={user1}},\n\labelify-replace=\{\{ and \}\{,\}\{% first substitution\n\{[ \string\-]\}\{% second substitution\n\{\glshex00D8\}\{0\}% third substitution\n\},\n\labelify-list={user1}\n]

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 fontspec was detected.
5.5 Field and Label Options

**label-prefix=⟨tag⟩**

The **label-prefix** option prepends ⟨tag⟩ to each entry’s label. This ⟨tag⟩ will also be inserted in front of any cross-references, unless they start with dual. or tertiary. or `ext⟨n⟩` (where ⟨n⟩ is an integer). Use **dual-prefix** to change the dual label prefixes and **ext-prefixes** to change the external label prefixes.

As from version 1.8, the primary label prefix is identified in the .glistex file with:

```latex
\bibglsprimaryprefixlabel{⟨prefix⟩}
```

For example, if the .bib file contains:

```plaintext
@entry{bird,
    name={bird},
    description = {feathered animal, such as a \gls{duck} or \gls{goose}}
}
```

```plaintext
@entry{waterfowl,
    name={waterfowl},
    description={Any \gls{bird} that lives in or about water},
    see={{[see also]{duck,goose}}}
}
```

```plaintext
@index{duck}
```

```plaintext
@index{goose,plural="geese"}
```

Then if this .bib file is loaded with **label-prefix={gls.}** it’s as though the entries had been defined as:

```plaintext
@entry{gls.bird,
    name={bird},
    description = {feathered animal, such as a \gls{gls.duck} or \gls{gls.goose}}
}
```

```plaintext
@entry{gls.waterfowl,
    name={waterfowl},
    description=={Any \gls{gls.bird} that lives in or about water},
    see={{[see also]{gls.duck,gls.goose}}}
}
```

```plaintext
@index{gls.duck,name={duck}}
```

```plaintext
@index{gls.goose,name={goose},plural="geese"}
```

Remember to use this prefix when you reference the terms in the document with commands like \gls.
**5.5 Field and Label Options**

**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 \glistex 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. This suffix is only set just before writing the entry definition to the \glistex 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.

**record-label-prefix=⟨tag⟩**

If set, this option will cause bib2gls to pretend that each record label starts with ⟨tag⟩, if it doesn’t already. For example, suppose the records in the .aux file are:

```latex
\glsxtr@record{bird}{}{page}{glsnumberformat}{1}
\glsxtr@record{waterfowl}{}{page}{glsnumberformat}{1}
\glsxtr@record{idx.duck}{}{page}{glsnumberformat}{1}
\glsxtr@record{idx.goose}{}{page}{glsnumberformat}{1}
```

The use of record-label-prefix={idx.} makes bib2gls act as though the records were given as:

```latex
\glsxtr@record{idx.bird}{}{page}{glsnumberformat}{1}
\glsxtr@record{idx.waterfowl}{}{page}{glsnumberformat}{1}
\glsxtr@record{idx.duck}{}{page}{glsnumberformat}{1}
\glsxtr@record{idx.goose}{}{page}{glsnumberformat}{1}
```

**cs-label-prefix=⟨tag⟩**

If you have commands such as \gls{⟨label⟩} or \glstext{⟨label⟩} in field values (in situations where nested link text won’t cause a problem) the ⟨label⟩ will be converted as follows:

- if ⟨label⟩ starts with dual. then dual. will be replaced by the dual-prefix value;
- if ⟨label⟩ starts with tertiary. then tertiary. will be replaced by the tertiary-prefix value;
- if ⟨label⟩ starts with ext⟨n⟩. then ext⟨n⟩. will be replaced by the corresponding ext-prefixes setting (if cross-resource reference mode is enabled, see section 1.4);
5.5 Field and Label Options

- If \langle label \rangle doesn’t start with one of the above recognised prefixes then, if \texttt{cs-label-prefix} has been used the supplied value will be inserted otherwise the \texttt{label-prefix} setting will be inserted.

For example, given:

\begin{verbatim}
@entry{bird,
    name={bird},
    description = {feathered animal, such as a \texttt{\gls{duck}} or \texttt{\gls{goose}}} }
\end{verbatim}

then if \texttt{label-prefix=\{idx.\}} is set but \texttt{cs-label-prefix} isn’t included in the resource option list this will convert the \texttt{description} field to:

\begin{verbatim}
description = {feathered animal, such as a \texttt{\gls{idx.duck}} or \texttt{\gls{idx.goose}}}
\end{verbatim}

However with \texttt{cs-label-prefix=\{gls.\}} the \texttt{description} field will be converted to:

\begin{verbatim}
description = {feathered animal, such as a \texttt{\gls{gls.duck}} or \texttt{\gls{gls.goose}}}
\end{verbatim}

regardless of the \texttt{label-prefix} setting. Whereas if the original entry definition is:

\begin{verbatim}
@entry{bird,
    name={bird},
    description = {feathered animal, such as a \texttt{\gls{dual.duck}} or \texttt{\gls{dual.goose}}} }
\end{verbatim}

then dual. will be replaced by the value of the \texttt{dual-prefix} option regardless of the \texttt{cs-label-prefix} setting.

The \texttt{cs-label-prefix} setting doesn’t affect labels in the fields that have an entry label or label list as the value (\texttt{parent}, \texttt{alias}, \texttt{see} and \texttt{seealso}).

\texttt{ext-prefixes=\langle list \rangle}

Any cross-references in the \texttt{.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 \texttt{ext\langle n\rangle} will simply be removed. The default setting is an empty list, which will strip all \texttt{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 \texttt{.glistex} file with:

\begin{verbatim}
\bibglsexternalprefixlabel{\langle n \rangle}{\langle \texttt{prefix} \rangle}
\end{verbatim}

For example, suppose the file \texttt{entries-terms.bib} contains:
5.5 Field and Label Options

@entry{set,
   name={set},
   description={collection of values, denoted \gls{ext1.set}}
}

and the file entries-symbols.bib contains:

@symbol{set,
   name={\ensuremath{\mathcal{S}}},
   description={a \gls{ext1.set}}
}

These files both contain an entry with the label set but the description field includes \gls{ext1.set} which is referencing the entry from the other file. These two files can be loaded without conflict using:

\usepackage[record,symbols]{glossaries-extra}

\GlsXtrLoadResources[src={entries-terms},
   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:

collection of values, denoted \gls{sym.set}

The set entry from entries-symbols.bib will be defined with the label sym.set and the description will be:

a \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 \gls{ext1.S}}
}
and `entries-symbols.bib` could contain:

```latex
@symb{:S,
  name={\ensuremath{\mathcal{S}}},
  description={a \gls{ext1.set}}
}
```

Now they can be combined with:

```latex
\GlsXtrLoadResources[src={entries-terms,entries-symbols}]  
```

which will simply strip the `ext1.` prefix from the cross-references. Alternatively:

```latex
\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.

### save-original-id=

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 `.glistex` 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=

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`.

140
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}
@index{coconut-matting,
   name={coconut matting}
}
@index{track}
@index{furrow}
```

and suppose the file entries-pt.bib contains:

```latex
@index{gato,
   prefix={o},
   translations-en={cat,staple,rivet},
   seealso={gatinho}
}
@index{gatinho,
   translations-en={kitten}
}
```

141
5.5 Field and Label Options

@index{tapete,
   translations-en={carpet,rug,mat,tapestry}
}

@index{esteira,
   prefix={a},
   translations-en={mat,track,mattting,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[english,brazilian]{babel}
\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},}
\begin{document}
\selectlanguage{british}
The \gls{cat} sat on the \gls{mat}.
\selectlanguage{brazilian}
0 \gls{gato} sentou-se no \gls{tapete}.
\end{document}

\renewcommand*{\glsxtrbookindexname}{1}{}
\glsxtrifhasfield{prefix}{1}{\xmakefirstuc\glscurrentfieldvalue\_}\{\%
\glossentryname{1}%
\glsxtrifhasfield{useri}{1}\{1\%
\glsxtrifhasfield{translations}{1}\{1\%
\}; translations: \glsxtrseelist\glscurrentfieldvalue\%
\}
\printunsrtglossaries
\end{document}

**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 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:
5.5 Field and Label Options

\GlsXtrLoadResources\[sort={integer},group={Constants},
\hspace{1em}src={entries-constants}% data in entries-constants.bib\
\GlsXtrLoadResources\[sort={letter-case},group={Variables},
\hspace{1em}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.

\textbf{category=}\textit{⟨value⟩}

The selected entries may all have their category field changed before writing their definitions to the .glistex file. The \textit{⟨value⟩} may be:

- \textbf{same as entry}: set the \textit{category} to the .bib entry type used to define it (lower case and without the initial @) after any aliasing, if applicable;

- \textbf{same as original entry}: (new to v1.4) set the \textit{category} to the original entry type (lower case and without the initial @) before it was aliased (behaves like \textbf{same as entry} if the entry type wasn’t aliased);

- \textbf{same as base}: (new to v1.1) set the \textit{category} to the base name of the .bib file (without the extension) that provided the entry definition;

- \textbf{same as type}: set the \textit{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);

- \textbf{⟨label⟩}: the \textit{category} is set to \textit{⟨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=\textbf{same as entry} refers to the target entry type whereas category=\textbf{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 \textit{save-original-entrytype=}\textit{category}. When combined with \textit{save-original-entrytype-action=}\textit{changed} it’s then possible to only set the \textit{category} to the original entry type for aliased entries and leave it unmodified for unaliased entries.

For example, if the .bib file contains:

@entry{bird,
\hspace{1em}name={bird},
\hspace{1em}description = {feathered animal}}

@index{duck}
5.5 Field and Label Options

@index{goose, plural="geese"}
@dualentry{dog,
    name={dog},
    description={chien}
}

then if the document contains:

\GlsXtrLoadResources[category={same as entry}, src={entries}]

this will set the \texttt{category} of the \texttt{bird} term to \textit{entry} (since it was defined with \texttt{@entry}), the \texttt{category} of the \texttt{duck} and \texttt{goose} terms to \textit{index} (since they were defined with \texttt{@index}), and the \texttt{category} of the \texttt{dog} term to \texttt{dualentry} (since it was defined with \texttt{@dualentry}). Note that the dual entry \texttt{dual.dog} doesn’t have the \texttt{category} set, since that’s governed by \texttt{dual-category} instead.

If, instead, the document contains:

\GlsXtrLoadResources[category={animals}, src={entries}]

then the \texttt{category} of all the primary selected entries will be set to \textit{animals}. Again the dual entry \texttt{dual.dog} doesn’t have the \texttt{category} set.

Note that the categories may be overridden by the commands that are used to actually define the entries (such as \texttt{\bibglsnewindex}).

For example, if the document contains:

\newcommand{\bibglsnewdualentry}[4]{%
\longnewglossaryentry*{#1}{name={#3},#2,category={dual}}{#4}%
}

\GlsXtrLoadResources[category={animals}, src={entries}]

then both the \texttt{dog} and \texttt{dual.dog} entries will have their \texttt{category} field set to \textit{dual} since the new definition of \texttt{\bibglsnewdualentry} has overridden the \texttt{category={animals}} option.

\texttt{type=⟨value⟩}

The \texttt{⟨value⟩} may be one of:

- \texttt{same as entry} set the \texttt{type} field to the entry type (lower case and without the initial \texttt{@});
- \texttt{same as original entry} set the \texttt{type} to the original entry type (lower case and without the initial \texttt{@}) before it was aliased (behaves like \texttt{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 \texttt{.bib} file (without the extension);
5.5 Field and Label Options

- **same as category**: set the *type* field to the same value as the *category* field (*type* unchanged if *category* not set);

- **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. Entries should always have the same type as their parent, but it’s possible for spawned entries to pick up the *type* field from their progenitor entry (if it was explicitly set in the `.bib` file), which may be inappropriate.

- `<label>` sets the *type* field to the glossary identified by `<label>`. When used with `entry-type-aliases`, the option `type={same as entry}` refers to the target entry type and `type={same as original entry}` refers to the original entry type given in the `.bib` file. An alternative is to use `save-original-entrytype={type}`. When combined with `save-original-entrytype-action={changed}` it’s then possible to only set the *type* to the original entry type for aliased entries and leave it unmodified for unaliased entries.

It’s not possible to have both *category*={same as type} and *type*={same as category}.

Note that this setting only changes the *type* field for primary entries. Use `dual-type` for dual entries.

For example:

\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.
5.5 Field and Label Options

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`). You can define the glossary before loading the resource, but it's not required as `bib2gls` will write `\provideignoredglossary*{⟨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`.

`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.

`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:
5.5 Field and Label Options

@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 .glistex 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 \langle field1\rangle=\langle field2\rangle pairs, where \langle field1\rangle and \langle field2\rangle are field names. Identical mappings and trails aren’t permitted. (That is, \langle field1\rangle and \langle field2\rangle can’t be the same nor can you have both \langle field1\rangle=\langle field2\rangle and \langle field2\rangle=\langle field3\rangle.) 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 \langle field1\rangle is listed in ignore-fields it won’t be ignored (unless \langle field2\rangle is in ignore-fields).

For example, suppose people.bib contains:

@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}
}

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}
]
replicate-fields=⟨key=value list⟩

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}
}

149
then:

\GlsXtrLoadResources[
  src={people},% data in people.bib
  replicate-fields={name=first}
]

won’t alter the \texttt{first} field since \texttt{replicate-fields} doesn’t override existing values by default. You can use \texttt{replicate-override} to change this. Alternatively, since \texttt{replicate-fields} is always performed after \texttt{ignore-fields} it’s possible to ignore the \texttt{first} field which means that the \texttt{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}
]

\texttt{replicate-override={⟨boolean⟩}}

This is a boolean option. The default setting is \texttt{replicate-override={false}}. If \texttt{true}, \texttt{replicate-fields} will override the existing value if the target field is already set.

\texttt{replicate-missing-field-action={⟨value⟩}}

This option indicates what to do if a source field identified in \texttt{replicate-fields} is missing. The value may be one of:

- \texttt{skip}: skip the replication of the missing field (default);
- \texttt{fallback}: use the fallback for the missing field, if one is available (otherwise skip);
- \texttt{empty}: make the target field empty.
5.5 Field and Label Options

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).

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}]\n
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}.

save-original-entrytype={value}

The ⟨value⟩ 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 ⟨value⟩ is omitted or the keyword true, then save-original-entrytype={originalentrytype} If ⟨value⟩
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 .glistex 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.

\texttt{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 \texttt{save-original-entrytype={false}} otherwise
the action is determined by \texttt{⟨value⟩} 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;
- changed override or changed or diff: only save the original entry type if it’s dif-
ferent 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 \texttt{ignore-fields}.

The “changed” options ignore case. For example, if the .bib file defined an entry with
\texttt{@INDEX} then both the original and final entry type will be index.

\textbf{Field Adjustments}

\texttt{post-description-dot=⟨value⟩}

The postdot package option (or \texttt{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 \texttt{⟨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.
5.5 Field and Label Options

The `post-description-dot={check}` setting determines whether to append the dot as follows:

- If the `description` field ends with `\nopostdesc` 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.
5.5 Field and Label Options

check-end-punctuation=\langle list\rangle

This options checks the end of all the fields given in \langle list\rangle for end of sentence punctuation. This is determined as follows, for each \langle field\rangle in the comma-separated \langle list\rangle:

- 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 \langle field\rangleendpunc 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 \langle list\rangle.

The default sentence.terminators is defined in bib2gls-en.xml as:

\<entry key="sentence.terminators">.?!</entry>?

Any character that isn’t of type “Punctuation, Other” won’t match.

For example, the sample books.bib file contains:

@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:

\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:
5.5 Field and Label Options

\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:

\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).

\texttt{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:
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 \texttt{sort} or \texttt{dual-sort}), if this option is set, then for each \{\texttt{\textbackslash{}\{field-list\}\texttt{\}}:\texttt{\langle\texttt{sort}\texttt{\}}:\texttt{\langle\texttt{csname}\texttt{\}}\texttt{\}} the following steps are performed:

1. For each entry \langle\texttt{id}\texttt{\}}:
   a) For each \langle\texttt{field}\texttt{\}} in \langle\texttt{field-list}\texttt{\}}, if the field is set for entry \langle\texttt{id}\texttt{\}} then:
      i. The field value must be in the form \{\langle\texttt{tag}\texttt{\}}\langle\texttt{label-list}\texttt{\}} where \{\langle\texttt{tag}\texttt{\}}\texttt{\}} is optional and \langle\texttt{label-list}\texttt{\}} is a comma-separated list of entry labels \langle\texttt{label}_{1}\texttt{\}}, \ldots, \langle\texttt{label}_{n}\texttt{\}};
      ii. A new list is constructed where the \textit{i}th element is: \{\langle\texttt{csname}\texttt{\}}\langle\texttt{label}_{i}\texttt{\}}\texttt{\}} unless \langle\texttt{csname}\texttt{\}} hasn’t been set, in which case the \textit{i}th element is just \{\langle\texttt{label}_{i}\texttt{\}}\texttt{\}} (the optional \{\langle\texttt{tag}\texttt{\}}\texttt{\}} part is omitted);
      iii. This new list is sorted according to the interpreter’s definition of the command given by \langle\texttt{csname}\texttt{\}} (if provided) and the designated \langle\texttt{sort}\texttt{\}} method;
      iv. The field value is reconstructed with the labels in the corresponding order (prefixed with \{\langle\texttt{tag}\texttt{\}}\texttt{\}} 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:

\begin{verbatim}
@index{bird}

@index{waterfowl, parent={bird} }

@index{duck, 
parent={waterfowl}, 
seealso={swan,duckling,parrot,goose} }

@index{swan, 
parent={waterfowl}, 
seealso={goose,duck} }

@index{goose, }
\end{verbatim}
5.5 Field and Label Options

parent={waterfowl},
seealso={duck}
}

@index{parrot, parent={bird} }

@index{duckling,
  see={\{related terms\}fluffy,velociraptor,duck,tardigrade}
}
@index{fluffy}
@index{tardigrade, name={water bear} }
@index{velociraptor}

And suppose the document contains:
\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 andseealso fields according to the referenced entry’s name (obtained with \glsentryname).

For example, the see field for the duckling entry was originally:
see={\{related terms\}fluffy,velociraptor,duck,tardigrade}

but in the .glistex file it’s written as:
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:
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:

```latex
seealso={swan,duckling,parrot,goose}
```

but in the `.glstex` file it’s written as:

```latex
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 `\gls-entryname`:

```latex
\GlsXtrLoadResources[
  src={entries},
  sort={en},
  sort-label-list={{seealso,see}:en:glsxtrhiername}
]
```

The separator between the levels is given by `\glsxtrhiernamesep` 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 `\glsxtrhiernamesswanswan` 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:

```latex
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 `\glsxtrhiername`:

```latex
\renewcommand*{\glsseeitemformat}[1]{\glsxtrhiername{#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:
5.5 Field and Label Options

@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 user1 field.

The document has two glossaries for each set of terms. The English terms are sorted according to sort={en-GB} in one resource set and the Portuguese terms are sorted according to sort={pt-BR} in another resource set. This means that there are cross-resource references, but since there are no instances of @preamble it should be possible to resolve the references.

The document code is:
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[record, noname, 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-aliasies={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}}

\begin{document}
\renewcommand*{\glsxtrbookindexname}[1]{%\glossentryname{#1}%
  \glsxtrifhasfield{useri}{#1}{: \glsxtrseelist\glscurrentfieldvalue}{%}
}\printunsrtglossaries
\end{document}

In verbose mode, the transcript file indicates that it’s performing the label list sorting. For example, when sorting according to \texttt{sort-label-list={user1:pt-BR:glsentryname}}, the transcript file contains:

\textbf{Label list sort method 'pt-BR' on field: user1}

The \texttt{cat} entry has a list of two elements in this field: \texttt{gato}, \texttt{gatinho}. This is converted into a new list where the first element is:

\texttt{\glsentryname{gato}}

and the second element is:

\texttt{\glsentryname{gatinho}}

Regardless of the level of verbosity, the transcript file will contain the conversions obtained by the interpreter:

texparserlib: \texttt{\glsentryname{gato}} -> gato
texparserlib: \texttt{\glsentryname{gatinho}} -> gatinho

The \texttt{kitten} entry has the same list, and the same process is repeated for that entry. The \texttt{--verbose} mode will provide additional information. The \texttt{--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.
5.5 Field and Label Options

\texttt{bibtex-contributor-fields=\langle list\rangle}

This option indicates that the listed fields all use \LaTeX{}'s name syntax (as used in \LaTeX{}'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{\bibglscontributorlist\langle\textit{contributor list}\rangle\{\langle n\rangle\}}

where \langle n \rangle is the number of names in the list and \langle contributor-list \rangle is a comma-separated list of names in the form:

\texttt{\bibglscontributor\langle\textit{forenames}\rangle\{\langle von-part\rangle\}\{\langle surname\rangle\}\{\langle suffix\rangle\}}

The \texttt{\bibglscontributorlist} 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{\bibglscontributorlist} provided in the .glstex file depends on whether or not \texttt{\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 \texttt{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 \texttt{name} field value to:

\texttt{\bibglscontributorlist\%
\bibglscontributor\{John\}\{Smith\}\{\%
\bibglscontributor\{Jane\}\{Doe\}\{\%
\bibglscontributor\{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{\bibglscontributor} command is defined in \texttt{bib2gls}'s interpreter and its definition is dependent on this setting. The \langle value \rangle may be one of (where the parts in square brackets are omitted if that argument is empty):

- \texttt{surname}: \texttt{\bibglscontributor} expands to \langle surname\rangle, \langle suffix\rangle, \langle forenames\rangle, \langle von-part\rangle;
5.5 Field and Label Options

- von: `\bibglscontributor` expands to `⟨von-part⟩⟨surname⟩[⟨suffix⟩][ ⟨forenames⟩];`
- forenames: `\bibglscontributor` expands to `⟨forenames⟩⟨von-part⟩⟨surname⟩[ ⟨suffix⟩][ ⟨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:

```latex
\newcommand*{\bibglscontributor}[4]{%#1\ifstrempty{#2}{}{ #2} #3\ifstrempty{#4}{}{, #4}%;}
```

\GlsXtrLoadResources[
  src={entries}, % data in entries.bib
  bibtex-contributor-fields={name}
]

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:

\GlsXtrLoadResources[
  src={no-interpret-preamble},
  interpret-preamble={false},
  bibtex-contributor-fields={name},
  contributor-order={forenames}
]

\GlsXtrLoadResources[
  src={entries}, % data in entries.bib
  bibtex-contributor-fields={name}
]

Note the need to use `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 `\bibglscontributor` is only written to the `.glstex` file if `bibtex-contributor-fields` has been set to a non-empty list. The second resource set will use the default `bibtex-contributor-fields={von}` setting when obtaining the sort value.

`encapsulate-fields={⟨key=value list⟩}`

This option should take a comma-separated list of `<field>=⟨cs-name-1arg⟩` values, where `<cs-name-1arg>` is the name of a control sequence that takes one argument. The value is required
5.5 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{interpret-fields=\{\langle list\rangle\}}

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 \texttt{sort} field, any special characters are replaced with commands like \texttt{\glshbackslash} and \texttt{\bibglsdollarchar}. This option is applied after \texttt{field-case-change} (if set).

For example, suppose I have a file \texttt{entries.bib} that contains definitions like:

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

163
5.5 Field and Label Options

@symbol{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:

\newcommand{\definition}{% 
  \ifglsentryexists{#1}% 
    {% 
      \section{\glsentryname{#1}}{\glsadd{#1}\glsxtrglossentry{#1}}% 
      \Glossentrydesc{#1}\glspostdescription
    }% 
    {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 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 hyperref:

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 bib2gls’ interpreter recognises these commands, I can get it to make the conversion instead of trying to implement a method within \TeX:

\glsaddstoragekey{pdfname}{}{\pdfname}
\GlsXtrLoadResources[
    src={entries},
    replicate-fields={name=pdfname},
    replicate-missing-field-action={fallback},
    interpret-fields={pdfname}
]

This first copies the name field to the custom pdfname and then interprets the copy. This leaves the name field with the \TeX code to produce the symbol in the document, but the 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 X\TeX or Lua\TeX these characters can be written to the PDF bookmarks by adjusting the definition of the custom command:
5.5 Field and Label Options

\newcommand{\definition}[1]{% 
  \ifglsentryexists{#1}% 
  {% 
    \section{\texorpdfstring{\glsentryname{#1}}{pdfname{#1}}}% 
    {\glsadd{#1}\glsxtrglossentry{#1}}% 
    {\Glossentrydesc{#1}\glspostdescription} \% 
  } \% 
  \section[Missing `#1']{\glsadd{#1}}}% 

With pdf\TeX{} 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=\{(value)\}}

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=\{(list)\}}

This option will convert any Unicode characters (outside of the Basic Latin set) that are found in the listed fields into \texttt{\bibglсхexunicodechar\{\langle hex-code\}\}} where \texttt{\langle hex-code\}} 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\}} instead of the Unicode character.
5.5 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:

\bibglst.datetime{⟨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:

\bibglst.date{⟨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:

\bibglst.time{⟨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.
5.5 Field and Label Options

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 \pgls 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 \hyphen 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.
prefix-fields=\langle list \rangle

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=\langle value \rangle

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=\langle cs \rangle

Identifies the command \langle cs \rangle 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=\langle sequence \rangle

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\langle hex \rangle 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=\langle sequence \rangle

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 \langle sequence \rangle 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.
append-prefix-field-nbsp-match=\{\texttt{pattern}\}

The value is the regular expression that identifies prefixes that should be followed by \texttt{~} instead of \texttt{space}. The default is \texttt{append-prefix-field-nbsp-match=\{.\}} which indicates a single character.

\section*{Case-Changing}

The \texttt{glossaries-extra} package comes with the category attributes \texttt{glossdesc} and \texttt{glossname}, which may take the values \texttt{firstuc} or \texttt{title}. These don't change the actual \texttt{name} or \texttt{description} fields, but instead \texttt{\glossentryname} and \texttt{\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 \texttt{\glossentryname} will use \texttt{\Glsentryname} if the \texttt{glossname} attribute for the given entry is set to \texttt{firstuc} and \texttt{\glossentrydesc} will use \texttt{\Glsentrydesc} if the \texttt{glossdesc} attribute is set to \texttt{firstuc}. The \texttt{title} setting will instead use \texttt{\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 \texttt{bib2gls} will perform the actual case-change according to its own algorithm. The results can vary according to the field content.

Only a subset of known fields have a resource option that can be used to apply a case-change. For example, \texttt{name-case-change} can be used to change the case of the \texttt{name} field, but there's no equivalent option for the \texttt{text} field.

Each of the case-changing resource options may take one of the following values:

- \texttt{none}: don't apply any case-changing (default);
- \texttt{lc-cs}: make \texttt{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 = \{\texttt{bibglslowercase}\{\langle text \rangle\}}
  \end{verbatim}

  which uses \texttt{\TeX} to convert the field to lower case;
- \texttt{uc-cs}: make \texttt{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 = \{\texttt{bibglsuperscase}\{\langle text \rangle\}}
  \end{verbatim}
which uses \TeX to convert the field to upper case;

- \texttt{firstuc-cs}: make \texttt{bib2gls} behave as though the field assignment:

\[
\langle \text{field} \rangle = \{ \langle \text{text} \rangle \}
\]

had actually been specified as:

\[
\langle \text{field} \rangle = \{ \texttt{\bibglsfirstuc}{\langle \text{text} \rangle} \}
\]

which uses \TeX to convert the field to first-letter upper case;

- \texttt{title-cs}: make \texttt{bib2gls} behave as though the field assignment:

\[
\langle \text{field} \rangle = \{ \langle \text{text} \rangle \}
\]

had actually been specified as:

\[
\langle \text{field} \rangle = \{ \texttt{\bibglstitlecse}{\langle \text{text} \rangle} \}
\]

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 \texttt{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 \texttt{mfirstuc-english}).

The \texttt{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.
5.5 Field and Label Options

The \texttt{firstuc-cs} and \texttt{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.

The \texttt{⟨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 \texttt{textcase} and \texttt{mfirstuc} package documentation for further details [1, 11]. You may use \texttt{\NoCaseChange{⟨content⟩}} (provided by \texttt{textcase}) to prevent any case-change to \texttt{⟨content⟩}.

For the settings where \texttt{bib2gls} itself performs the case-change, then \texttt{bib2gls} will iterate over each token of the field value and apply the following rules:

1. If the token is a normal Unicode alphabetic character, it will be replaced with the corresponding upper or lower case character, as appropriate. For \texttt{title} and \texttt{firstuc}, the title case character is used as the replacement, for \texttt{uc} the upper case character is used as the replacement, and for \texttt{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 \texttt{title}, \texttt{firstuc} and \texttt{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 \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.

2. If the token is a normal Unicode character that isn’t alphabetical, then this token will be skipped for all options.

3. If \texttt{$⟨maths⟩$} is encountered, it will be skipped. If the option is \texttt{firstuc} then all remaining tokens are skipped, and no case-change will be performed.

4. If a group \texttt{⟨content⟩} is found, then the case-change is applied to the entire \texttt{⟨content⟩} (which may be empty). This corresponds to the way \texttt{\makefirstuc} and \texttt{\capitalisewords} work if a word starts with a group. Note that with \texttt{firstuc} and \texttt{title} the group content will be converted according to \texttt{uc}, so the normal upper case character is used rather than the title case character (if they are different).

   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.

5. If a control sequence \texttt{\⟨csname⟩} is found, then:

   a) If the control sequence is \texttt{\protect}, this token is skipped for all options.

   b) If the control sequence is one of: \texttt{\o, \O, \i, \L, \ae, \AE, \oe, \OE, \aa, \AA, \ss, \SS, \ng, \NG, \th, \TH, \dh, \DH, \dj} or \texttt{\DJ}, then it’s replaced with its case-change counterpart (if not already the correct case).

   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.
5.5 Field and Label Options

c) If the control sequence is `\NoCaseChange` or is in the `no-case-change-cs` list, then the control sequence and its argument is ignored. With `firstuc` and `title`, if `\NoCaseChange{(text)}` (but not any `no-case-change-cs` command) 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). This is different to the way `\makefirstuc` and `\capitalisewords` work.

d) If the control sequence is `\ensuremath`, `\si` or if `(csname)` ends with “ref” (for example, `\ref` or `\pageref`) then the control sequence and its argument is ignored. In the case where `(csname)` ends with “ref”, a following star (*) or optional argument before the mandatory argument will also be skipped. This allows for some common cross-referencing commands, such as hyperref’s `\autoref`, which may have a starred form, but does not allow for more complicated commands with multiple arguments.

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 `\glsentrytitlecase` then:

- `lc` the control sequence is converted to `\glsxtrusefield`;
- `uc` the control sequence is converted to `\GLSxtrusefield`;
- `firstuc` the control sequence is converted to `\Glsxtrusefield` and the remaining tokens are skipped;
- `title` the control sequence is left unchanged and subsequent tokens are skipped until a word-boundary is found.

The field and entry label arguments are skipped.

f) If the control sequence is `\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 `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.

g) If the control sequence is `\glsdisp`, `\glslink`, `\dglsdisp` or `\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 `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.

h) If the control sequence has a known case variant, it will be substituted. For example, `\gls` will be changed to `\Gls` or `\GLS`. In some cases there isn’t an appropriate variant. For example, `\glsentrytext` has a first-letter upper case version `\Glsentrytext`, but not an all-caps version.
5.5 Field and Label Options

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.

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 `firstuc` and `title`. (Again, this follows the way that \makefirstuc and \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 `title` setting as they will be skipped by the `firstuc` setting.) For complex cases, consider using a semantic command that hides non-textual context such as the \strong example described on page 103.

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:

```latex
\GlsXtrLoadResources[
  name-case-change={uc},
  description-case-change={title}
]
```

won’t have an effect. Although the default long–short abbreviation style sets the \texttt{name} and \texttt{description} fields, \texttt{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:

```latex
\GlsXtrLoadResources[
  name-case-change={uc},
  description-case-change={title},
  replicate-fields={short=name,long=description}
]
```

then \texttt{bib2gls} will act as though the entry had been defined as:

```latex
@abbreviation{html,
  short = \{html\},
  long = \{hypertext markup language\},
  name = \{HTML\},
  description = \{Hypertext Markup Language\}
}
```

If the long–short-sc abbreviation style is set (before \texttt{\GlsXtrLoadResources}) then this will override the default style for the \texttt{name} and \texttt{description}, so \texttt{gls\{html\}} will display the short form using \textsc{html} but the \texttt{name} in the glossary will be displayed using just HTML.

Note that with \texttt{@index} the \texttt{name} and \texttt{text} fields will automatically be created if they are missing and \texttt{name-case-change} is used. For example, if an entry is defined as:
then \texttt{name-case-change=\{firstuc\}} will make this entry behave as though it was defined as:

```latex
@index{duck,
    name = \{Duck\},
    text = \{duck\}
}
```

Suppose I have a slightly eccentric abbreviation definition:

```latex
@abbreviation{html,
    short = "ht\textit{ml}" ,
    long = "hypertext markup language"
}
```

then \texttt{short-case-change=\{uc\}} would convert the value of the \texttt{short} field into:

\texttt{HT\textit{ML}}

Note that \textit{ 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:

```latex
@abbreviation{html,
    short = "{ht}ml",
    long = "hypertext markup language"
}
```

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:

```latex
@abbreviation{html,
    short = "\textit{ht}ml",
    long = "hypertext markup language"
}
```

then \texttt{short-case-change=\{firstuc\}} will convert the \texttt{short} field value to:

\texttt{\textit{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:

```latex
name = "z\ae\oe",
175
5.5 Field and Label Options

then with name-case-change={uc}, the value would be converted to

\text{Z\AE\OE}

because \ae and \oe have known upper case versions.

With name-case-change={uc-cs}, the name value would be converted to:

\text{\bibgluppercase{z\AE\OE}}

If the interpreter is used during sorting, the sort value will be set to Z\AE\OE because the interpreter recognises all three commands.

You can use \text{\NoCaseChange{⟨text⟩}} to prevent the given ⟨text⟩ from having the case changed. For example, if the short field is defined as:

\text{short = \{a\NoCaseChange{bc}d\}}

then with short-case-change={uc}, this would be converted to

\text{A\NoCaseChange{bc}D}

Note that with firstuc and title, if \NoCaseChange{⟨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 short field is defined as:

\text{short={\NoCaseChange{h}tml}}

then the result is:

\text{\NoCaseChange{h}Tml}

whereas with:

\text{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 short field in the .bib definition is defined as:

\text{short = "ht\textcolor{red}{ml}"},

then with short-case-change={uc} this would end up as:

\text{HT\textcolor{RED}{ML}}

which is incorrect. Instead, provide a command that hides the label (such as the \textbf{strong} example described on page 103).
5.5 Field and Label Options

no-case-change-cs=\{list\}

Instructs the non-T\TeX case-changing options (where \texttt{bib2gls}, not \TeX, performs the modification) to treat the commands whose control sequence names are given in the comma-separated \(\langle\text{list}\rangle\) in the same way as it treats \texttt{\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:

\begin{verbatim}
no-case-change-cs=\{fieldfmt,abbrstylefmt,glostylefmt\}
\end{verbatim}

word-boundaries=\{list\}

Governs how the title case-change option determines word boundaries. The \(\langle\text{list}\rangle\) 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 \texttt{\space} or \texttt{\␣} indicate a word-boundary;

dash a Unicode character that belongs to the “Punctuation, Dash” block indicates a word-boundary;

\texttt{\~} 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 \texttt{word-boundaries} within the same set of resource options will override previous settings.

The default setting is \texttt{word-boundaries=\{white space,cs space\}}, which excludes non-breakable spaces and dashes.

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) unless the entry type is @abbreviation or @acronym, in which case if the name field is missing no action is performed.

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 ⟨field⟩=⟨setting⟩ 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 ⟨setting⟩ 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) 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.
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).

5.6 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 $\text{gls\{⟨label⟩\}}[s]$ will sometimes produce inappropriate results, so the glossaries package provides a plural key with the corresponding command $\text{glspl}$.

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 $\text{glspluralsuffix}$ 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:

- If firstplural is missing, then $\text{glspluralsuffix}$ 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 $\text{glspluralsuffix}$ appended.

- If the longplural field is missing, then $\text{glspluralsuffix}$ 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, $\text{acrpluralsuffix}$ 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 $\text{abbrvpluralsuffix}$ 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, $\text{abbrvpluralsuffix}$ is redefined. Most styles simply define this command as:

\begin{verbatim}
\renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}
\end{verbatim}

where $\text{glsxtrabbrvpluralsuffix}$ expands to $\text{glspluralsuffix}$. The “sc” styles (such as long-short-sc) use a different definition:

\begin{verbatim}
\renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}
\end{verbatim}

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 $\text{abbrvpluralsuffix}$, as the change will be undone the next time the style is applied. Instead, for a document-wide solution, you need to redefine $\text{glsxtrabbrvpluralsuffix}$. Alternatively you can use the category attributes.
5.6 Plurals

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:

```latex
\glssetcategoryattribute{french}{noshortplural}
```

Now just make sure all the French abbreviations are have their `category` field set to `french`:

```latex
\GlsXtrLoadResources[src={fr-abbrvs},category={french}]
```

The other approach is to use the options listed below for the given resource set. For example:

```latex
\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.
5.7 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.

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 encapsulating command (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{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}

for standard the internal locations,

\glsxtrdisplaysupploc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨src⟩}{⟨location⟩}

for supplemental (external) locations and

\glsxtrdisplaylocnameref{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨title⟩}
{⟨href⟩}{⟨hcounter⟩}{⟨file⟩}

for nameref records. (See section 5.8 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 ⟨format⟩ 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\{xx\} 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\{xx\} commands.

Ranges can be explicitly formed using the parenthetical syntax \textit{format={()} and format=\{() and format=}\{()\} (where \textit{csname} is again the name of a text-block command without the initial backslash) in the optional argument of commands like \gls or \glsadd. These will always form a range, regardless of \textit{min-loc-range}, and will be encapsulated by \bibglsrange. (This command is not used with ranges that are formed by collating consecutive locations.) The initial marker is stripped from the \textit{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, but will absorb any single locations within the range that don’t conflict. (Conflicts will be moved to the start of the explicit range.) For example, if \gls{sample} is used on page 1, \gls[format={}]{sample} is used on page 2, \gls{sample} is used on page 3, and \gls[format={}]{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 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[format={textbf}]{sample} then it would cause a warning and be moved before the start of the range so that the location list would then be 1, 3, 2–4.

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 \textit{format={glsignore}} is provided by the glossaries package for cases where the location should be ignored. (The command \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 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 \glsaddallunused but remember that iterative commands like this don’t work with bib2gls. Instead, just use \textit{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:

\backmatter
\GlSXtrSetDefaultNumberFormat{glsignore}

If you also want to drop front matter locations as well:
Note that explicit range formations aren’t discarded, so if \texttt{glsignore} is used in a range, such as:

\begin{verbatim}
glsadd[format={(glsignore}]{sample}
\end{verbatim}

then the range will be included in the location list (encapsulated with \texttt{glsignore}), but this case would be a rather odd use of this special format and is not recommended.

The record counting commands, such as \texttt{\rgls}, use the special format \texttt{glstriggerrecordformat}, which \texttt{bib2gls} also treats as an ignored record and the same rules as for \texttt{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 \texttt{xindy} and \texttt{makeindex} where you can specify the ordering (such as lower case Roman first, then digits, etc), but unlike those applications, \texttt{bib2gls} allows any location, although it may not be able to work out an integer representation. (With \texttt{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 \texttt{\glossentry} (and the child form \texttt{\subglossentry}) ignore the final argument (which will be the \texttt{location} field) and instead parse the \texttt{loclist} field and re-order the locations or process them in some other way. Remember that you can also use \texttt{\glsnoidxloclist} provided by \texttt{glossaries}. For example:

\begin{verbatim}
glsfieldfetch{gls.sample}{loclist}\% fetch location list
\glsnoidxloclist{loclist}\% iterate over locations
\end{verbatim}

This uses \texttt{\glsnoidxloclisthandler} as the list’s handler macro, which simply displays each location separated by \texttt{\delimN}. (See also Iteration Tips and Tricks [16].)

Each regular location is listed in the .aux file in the form:

\begin{verbatim}
\glsxtr@record{\langle label\rangle}{\langle prefix\rangle}{\langle counter\rangle}{\langle format\rangle}{\langle location\rangle}
\end{verbatim}

(See \texttt{--merge-nameref-on} for \texttt{nameref} records.) Exact duplicates are discarded. For example, if cat is indexed twice on page 1:

\begin{verbatim}
\glsxtr@record{cat}{}{page}{glsnumberformat}{1}
\glsxtr@record{cat}{}{page}{glsnumberformat}{1}
\end{verbatim}
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:

\glsxtr@record{cat}{{page}}{glsnumberformat}{1}
\glsxtr@record{cat}{{page}}{hyperbf}{1}

This is a partial record match. In this case, bib2gls makes the following tests:

- If one of the formats includes a range formation, the range takes precedence.
- If one of the formats is globsnumberformat (as in the above example) or an ignored record format such as g1signore, 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 globsnumberformat will take precedence over the ignored record formats (g1signore and glostriggerrecordformat).
- 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 seesalso 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=⟨boolean⟩

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 true or false. If true then bib2gls won’t save the location or loclist fields.

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.
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 .glistex file.

**save-loclist**\langle boolean\rangle

If you want the location field but don’t need loclist, you can use `save-loclist={false}`. This can help to save resources and build time.

**save-primary-locations**\langle value\rangle

It’s sometimes useful to identify primary locations 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 primary location 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 primary locations with `primary-location-formats` and use `save-primary-locations` to determine how to deal with these locations.

This option may take one of the following values:

- **false**: don’t save primary locations (default);
- **retain**: save primary locations in the `primarylocations` field but don’t remove from the usual location list;
- **default format**: similar to retain but the format for the primary records in the location field is converted to the default `glsnumberformat` ENCAP (the records in the `primarylocations` field retain their given format);
- **start**: save primary locations in the `primarylocations` field and also move to the start of the usual location list;
- **remove**: save primary locations in the `primarylocations` field and remove from the usual location list.

The primary locations are copied to the `primarylocations` field and encapsulated with `\bibglsp`. If you use `save-primary-locations={remove}`, the location field will end up empty if the locations for the associated entry were all identified as primary. If you use `save-primary-locations={start}`, all primary 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-primary-locations={remove}` and adjust the location list format to insert the primary 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:

\renewcommand*{\glsxtrbookindexprelocation}[1]{%
 \glsxtrifhasfield{primarylocations}{#1}%
}
5.7 Location List Options

(Notethatif loc-prefixis used, the prefix will be in the location field and so will come after the primary 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:

\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}

This option will automatically set save-primary-locations={retain} unless it has already been changed from the default save-primary-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 primary 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,
5.7 Location List Options

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}, 
 primary-location-formats={hyperbf,hyperemph}, 
 save-primary-locations={remove} ] 
\renewcommand*{\glsxtrbookindexprelocation}{\glstextrifhasfield{primarylocations}{#1}{% 
  \glstextrprelocation 
  \glstcurrentfieldvalue 
  \glstextrifhasfield{location}{#1}{;}{;}% 
}{} 
\glstextrprelocation } 
\glsxtrnewglslike[format={hyperbf}]{\primary}{\primarypl}{\Primary}{\Primarypl} 
\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}. 
\end{document}
5.7 Location List Options

\chapter{Yet Another Sample Again}
\Gls{waterfowl}, \gls{bird}, \primarypl{parrot} and \gls{zebra}.

\printunsrtglossary*[style={tree},nonumberlist]{%
\renewcommand*{\glsextrapostnamehook}[1]{\glsadd[format={hyperemph}]{#1}}%
}\printunsrtglossary*[title={Index},target={false}]
\end{document}

The primary-location-formats={hyperbf,hyperemph} setting in the above indicates that locations encapsulated with \hyperbf and \hyperemph are primary records. In this case, the bold format is used to indicate the primary location in the main document text and the emphasized format is used to indicate the location in the main glossary.

The primary records are removed from the location field due to the save-primary-locations={remove} setting. This can lead to a ragged location list. The option save-primary-locations={default format} can allow the primary 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 document build requires an extra \bib2gls and \LaTeX run:

pdflatex test
bib2gls --group test
dflatex test
bib2gls --group test
dflatex test

For consistency, I've used \glsxtrnewglslike to provide commands used to indicate a primary reference in the text. This means that if I decide to change the optional arguments used for primary references I only need to edit one line. For example, I might want to change the default counter:

\glsxtrnewglslike[format={hyperbf},counter={chapter}]{}{\primary}
{\primarypl}{\Primary}{\Primarypl}

Here's another example that only has one primary format (\hyperrm) that's indexed through the use of \GlsXtrAutoAddOnFormat, which sets up a hook that automatically inserts:

\glsadd[counter={chapter},format={hyperrm}]{(label)}

on each instance of \gls[format={primaryfmt}]{(label)} (or similar). This means that the entry is indexed twice when this particular format is used: first with the \hyperrm format and chapter counter (from the \glsadd command in the hook), and then with the primaryfmt format and the default counter (as per normal behaviour):
5.7 Location List Options

\documentclass{report}
\usepackage[colorlinks]{hyperref}
\usepackage[
    record={nameref},
    postpunc={dot},
    nostyles,
    stylemods={tree,bookindex},
    style={bookindex}]{glossaries-extra}
\GlsXtrLoadResources[
    src={topics},
    primary-location-formats={hyperrm},
    save-primary-locations={remove},
    save-loclist={false}]
\newcommand{\primaryfmt}[1]{\hyperbf{#1}}
\GlsXtrAutoAddOnFormat{primaryfmt}{counter={chapter},format={hyperrm}}
\glsxtrnewglslike[format={primaryfmt}]{\primary}{\primarypl}{\Primary}{\Primarypl}
\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*[\styles={tree},\title={Summary}]%
\renewcommand*{\glsextrapostnamehook}[1]{\glssadd[format={hyperemph}]\{#1\}}%
\renewcommand{\GlsXtrLocationField}{primarylocations}%
\printunsrtglossary[\title={Index},\target={false}]

5.7 Location List Options

Note that in this case, from bib2gls’ point of view, the primary format is hyperrm not primaryfmt. This picks out the records created with the automated \glsadd, which have the counter set to chapter. The first glossary (with the title “Summary”) switches the location field to primarylocations so that only the primary records are listed. Since 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 \glsadd records with the chapter counter have been removed because they were identified as primary records). These just show the page number as that’s the default display with record={nameref} for records with the page counter.

An alternative to \GlsXtrAutoAddOnFormat would be to simply define the custom commands as follows:

\newcommand{\primary}[2][]{%
  \glsadd[counter=chapter,format=hyperm]{#2}%
  \gls[format=primaryfmt]{#2}%
}\
\newcommand{\primarypl}[2][]{%
  \glsadd[counter=chapter,format=hyperm]{#2}%
  \glspl[format=primaryfmt]{#2}%
}\
\newcommand{\Primary}[2][]{%
  \glsadd[counter=chapter,format=hyperm]{#2}%
  \Gls[format=primaryfmt]{#2}%
}\
\newcommand{\Primarypl}[2][]{%
  \glsadd[counter=chapter,format=hyperm]{#2}%
  \Glspl[format=primaryfmt]{#2}%
}

This is more useful if you want to simply omit the format=primaryfmt option (just remove it from the above four definitions), which makes it easier to merge the locations into ranges in the index.

*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⟩. Otherwise the locations are separated by \bibglsdelimN or \bibgls lastDelimN. As mentioned above, these aren’t merged with explicit range formations.

You can change this 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.
5.7 Location List Options

bib2gls determines if one location \{\langle prefix-2\rangle\}\{\langle counter-2\rangle\}\{\langle format-2\rangle\}\{\langle location-2\rangle\} is one unit more than another location \{\langle prefix-1\rangle\}\{\langle counter-1\rangle\}\{\langle format-1\rangle\}\{\langle location-1\rangle\} according to the following:

1. If \langle prefix-1\rangle is not equal to \langle prefix-2\rangle or \langle counter-1\rangle is not equal to \langle counter-2\rangle or \langle format-1\rangle is not equal to \langle format-2\rangle, then the locations aren’t considered consecutive.

2. If either \langle location-1\rangle or \langle location-2\rangle are empty, then the locations aren’t considered consecutive.

3. If both \langle location-1\rangle and \langle location-2\rangle match the pattern (line break for clarity only):\n
\begin{verbatim}
(.*?)(?:\protect\s*)?(\[[p\{javaAlphabetic}\]+)\{([p\{javaDigit\}\p\{javaAlphabetic\}]*)\}
\end{verbatim}

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 \langle location-1\rangle set to group 1 of the first pattern match and \langle location-2\rangle set to group 1 of the second pattern match;
- otherwise the test is retried with \langle location-1\rangle set to group 3 of the first pattern match and \langle location-2\rangle set to group 3 of the second pattern match.

4. If both \langle location-1\rangle and \langle location-2\rangle match the pattern

\begin{verbatim}
(.*?)([^p\{javaDigit\}]?)(p\{javaDigit\}+)
\end{verbatim}

then:

a) if group 3 of both pattern matches are equal then:

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 location-1\rangle set to the concatenation of the first two groups \langle group-1\rangle\langle group-2\rangle of the first pattern match and \langle location-2\rangle set to the concatenation of the first two groups \langle group-1\rangle\langle group-2\rangle of the second pattern match;
iii. if the separators (group 2) are the same the test is retried with \langle location-1\rangle set to the first group \langle group-1\rangle of the first pattern match and \langle location-2\rangle set to the first group \langle group-1\rangle of the second pattern match.

\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.}
5.7 Location List Options

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 locations aren’t considered consecutive.

Examples:

1. \glsxtr@record{gls.sample}{}{page}{glsnumberformat}{1} \glsxtr@record{gls.sample}{}{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. \glsxtr@record{gls.sample}{}{page}{glsnumberformat}{1} \glsxtr@record{gls.sample}{}{page}{textbf}{2}

   These records aren’t consecutive since the formats are different.

3. \glsxtr@record{gls.sample}{}{page}{glsnumberformat}{A.i} \glsxtr@record{gls.sample}{}{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. \glsxtr@record{gls.sample}{}{page}{glsnumberformat}{i.A} \glsxtr@record{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.
5.7 Location List Options

The test fails because the numbers are equal and the prefixes are different.

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.

These records aren’t consecutive as the test branches off into step 4(a)ii.

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 \@alph{⟨number⟩} instead of fully expanding according to the normal behaviour of \alph{⟨counter⟩}. (Note that \glsxtrresourcefile changes the category code of @ to allow for internal commands in locations.) This unusual case is for illustrative purposes.

**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, ⟨location-1⟩ and ⟨location-2⟩, that have numeric values \(n_1\) and \(n_2\) (and identical prefix, counter and format), then the sequence ⟨location-1⟩, ⟨location-2⟩ is considered consecutive if

\[0 < n_2 - n_1 \leq ⟨\text{max-loc-diff}⟩\]

The default value of 1 means that ⟨location-2⟩ immediately follows ⟨location-1⟩ if \(n_2 = n_1 + 1\).

For example, if ⟨location-1⟩ is “B” and ⟨location-2⟩ is “C”, then \(n_1 = 66\) and \(n_2 = 67\). Since \(n_2 = 67 = 66 + 1 = n_1 + 1\) then ⟨location-2⟩ immediately follows ⟨location-1⟩.

This is used in the 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 location ranges that have skipped gaps in this manner will be followed by \bibglspassim. 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 \( \text{max-loc-diff} = \{2\} \) 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.

**suffix**F=(value)

If set, a range consisting of two consecutive locations \( \langle \text{loc-1} \rangle \) and \( \langle \text{loc-2} \rangle \) will be displayed in the location list as \( \langle \text{loc-1} \rangle (\text{value}) \).

Note that \( \text{suffixF} = \{} \) sets the suffix to the empty string. To remove the suffix formation use \( \text{suffixF} = \{\text{none}\} \).

The default is \( \text{suffixF} = \{\text{none}\} \).

**suffix**FF=(value)

If set, a range consisting of three or more consecutive locations \( \langle \text{loc-1} \rangle \) and \( \langle \text{loc-2} \rangle \) will be displayed in the location list as \( \langle \text{loc-1} \rangle (\text{value}) \).

Note that \( \text{suffixFF} = \{} \) sets the suffix to the empty string. To remove the suffix formation use \( \text{suffixFF} = \{\text{none}\} \).

The default is \( \text{suffixFF} = \{\text{none}\} \).

**compact-ranges**=(value)

The \( \langle \text{value} \rangle \) may be an integer \( \langle n \rangle \) or false (equivalent to \( \text{compact-ranges} = \{0\} \)) or true (equivalent to \( \text{compact-ranges} = \{3\} \)). If no \( \langle \text{value} \rangle \) 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 \( \text{\textbackslash bibglscompact} \), so the range would actually become:

\[ 184 \text{ \textbackslash delimR} \text{\bibglscompact\{digit\}\{18\}\{9\}} \]

If the location is in the form \( \langle \text{cs}\rangle\{\langle \text{loc} \rangle \} \) (where \( \langle \text{cs} \rangle \) is a command) then \( \text{\bibglscompact} \) will be inside the argument. For example, if the range would normally be:

\[ \text{\textbackslash custom}\{184\} \text{ \textbackslash delimR} \text{\textbackslash custom}\{189\} \]

then it would become:

\[ \text{\textbackslash custom}\{184\} \text{ \textbackslash delimR} \text{\textbackslash custom}\{\text{\textbackslash bibglscompact\{digit\}\{18\}\{9\}}\} \]

The numerical value given in \( \text{compact-ranges} = \{\langle n \rangle \} \) indicates that compaction should only occur if the actual location consists of at least \( \langle n \rangle \) characters, for \( \langle n \rangle \geq 2 \). Any value of \( \langle n \rangle \) less than 2 will switch off compaction.

For example, 189 consists of 3 characters, so it will be compacted with \( \text{compact-ranges} = \{3\} \) but not with \( \text{compact-ranges} = \{4\} \). Whereas \( \text{\textbackslash custom}\{89\} \) would only be compacted with \( \text{compact-ranges} = \{2\} \) 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 \bibglscompact 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:

- \texttt{omit}: omit the see reference from the location list.
- \texttt{before}: place the see reference before the location list.
- \texttt{after}: place the see reference after the location list (default).

The separator between the location list and the cross-reference is provided by \bibgls-see-seep. This separator is omitted if the location list is empty. The cross-reference is written to the location field using \bibglsusesee{⟨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 \bibglsseealso-seep. The cross-reference is written to the location field using \bibglsuseseealso{⟨label⟩}.

\textbf{alias=⟨value⟩}

This is like alias but governs the location of the cross-references provided by the alias field. The separator is given by \bibglsaliass-seep. The cross-reference is written to the location field using \bibglsusealias{⟨label⟩}.

\textbf{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:

- \texttt{keep}: keep the location list;
- \texttt{transfer}: transfer the location list;
- \texttt{omit}: omit the location list.
5.7 Location List Options

The default setting is \texttt{alias-loc={transfer}}. In all cases, the target entry will be added to the \texttt{see} field of the entry with the \texttt{alias} field, unless it already has a \texttt{see} field (in which case the \texttt{see} value is left unchanged).

Note that with \texttt{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 \texttt{glsxtrresourcefile}.) If you have \texttt{glossaries-extra} version 1.12, you may need to redefine \texttt{glsxtrsetaliasnoindex} to do nothing if the location lists aren’t showing correctly with aliased entries. (This was corrected in version 1.13.)

\textbf{loc-prefix=\langle value \rangle}

The \texttt{loc-prefix} setting indicates that the location lists should begin with \texttt{\bibglslocprefix\langle n\rangle}. The \langle value \rangle may be one of the following:

- \texttt{false}: don’t insert \texttt{\bibglslocprefix\langle n\rangle} at the start of the location lists (default).
- \texttt{\langle prefix-1\rangle,\langle prefix-2\rangle,...,\langle prefix-n\rangle}: insert \texttt{\bibglslocprefix\langle n\rangle} (where \langle n\rangle is the number of locations in the list) at the start of each location list and the definition of \texttt{\bibglslocprefix} will be appended to the glossary preamble providing an \texttt{\ifcase} condition:

\begin{verbatim}
\providecommand{\bibglslocprefix}[1]{% 
  \ifcase#1 \or \langle prefix-1\rangle\bibglspostlocprefix \\
  \or \langle prefix-2\rangle\bibglspostlocprefix \\
  \else \langle prefix-n\rangle\bibglspostlocprefix \fi }
\end{verbatim}

- \texttt{comma}: equivalent to \texttt{loc-prefix=\{, \}} but avoids confusion with the list syntax. That is, the prefix is a comma followed by a space for non-empty locations.
- \texttt{list}: equivalent to \texttt{loc-prefix=\pagelistname}.
- \texttt{true}: equivalent to \texttt{loc-prefix=\{\bibglspagename,\bibglspagesname\}}, where the definitions of \texttt{\bibglspagename} and \texttt{\bibglspagesname} are obtained from the tag, \texttt{page} and tag, \texttt{pages} entries in \texttt{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 value \rangle is omitted, \texttt{true} is assumed. Take care not to mix different values of \texttt{loc-prefix} for entries for the same \texttt{type} setting. It’s okay to mix \texttt{loc-prefix=\{false\}} with another value, but don’t mix non-\texttt{false} values. See the description of \texttt{\bibglslocprefix} for further details.

For example:
5.7 Location List Options

\GlsXtrLoadResources[type={main},src={entries1},loc-prefix={false}]
\GlsXtrLoadResources[type={main},src={entries2},loc-prefix]
\GlsXtrLoadResources[type={symbols},src={entries3},loc-prefix={p.,pp.}]

This works since the conflicting `loc-prefix={p.,pp.}` and `loc-prefix={true}` are in different glossaries (assigned through the `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 `type` option isn’t the same as setting the `type` field for each entry in the `.bib` file.)

If the `type` option isn’t used:

\GlsXtrLoadResources[src={entries1},loc-prefix={false}]
\GlsXtrLoadResources[src={entries2},loc-prefix]
\GlsXtrLoadResources[src={entries3},loc-prefix={p.,pp.}]

then `loc-prefix={true}` takes precedence over `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.

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}.

As with loc-prefix, take care not to mix different values of loc-suffix for entries in the same glossary type.

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}
This results in the location list “1, 1–3, 3–5”. This looks a little odd and it may seem as though the range formation hasn’t worked, but the locations are actually: page 1, equation 1, equation 2, equation 3, page 3, page 4 and page 5. Ranges can’t be formed across different counters.

The \texttt{loc-counters=\{list\}} option instructs \texttt{bib2gls} to group the locations according to the counters given in the comma-separated \texttt{\{list\}}. If a location has a counter that’s not listed in \texttt{\{list\}}, then the location is discarded.

For example:
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.

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⟩}
```
5.7 Location List Options

The location here is actually the value of the wrglossary counter not the page number, but bib2gls can pick up the corresponding \textit{page} from the \texttt{newlabel} command. It then replaces the record’s location \texttt{$\langle n \rangle$} with:

\begin{verbatim}
\glsxtr@wrglossarylocation{$\langle n \rangle$}{$\langle page \rangle$}
\end{verbatim}

(but it only does this for records that have the wrglossary counter).

The glossaries-extra package (v1.29+) adjusts the definition of \texttt{\glshypernumber} (which is internally used by \texttt{\glsnumberformat}, \texttt{\hyperbf} etc when hyperref has been loaded) so that if the counter is wrglossary then \texttt{\pageref} is used instead of \texttt{\hyperlink}. This means that the page number is displayed in the location list but it links back to the place where the corresponding \texttt{\label} occurred.

This method works partially with \texttt{makeindex} and \texttt{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 page number instead.

With the default \texttt{--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 \texttt{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 \texttt{indexcounter} internal field using:

\begin{verbatim}
\GlsXtrSetField{$\langle id \rangle$}{indexcounter}{\glsxtr@wrglossarylocation{$\langle n \rangle$}{\langle page \rangle}}
\end{verbatim}

Since \texttt{\glsxtr@wrglossarylocation} simply expands to its first argument, the corresponding label can be obtained with:

\begin{verbatim}
wrglossary.\glsxtr@wrglossarylocation{$\langle n \rangle$}{$\langle page \rangle$}
\end{verbatim}

For convenience, glossaries-extra-bib2gls provides:

\begin{verbatim}
\GlsXtrIndexCounterLink{$\langle text \rangle$}{$\langle label \rangle$}
\end{verbatim}

which will do:

\begin{verbatim}
\hyperref[wrglossary.\langle value \rangle]{$\langle text \rangle$}
\end{verbatim}

where \texttt{$\langle value \rangle$} is the value of the \texttt{indexcounter} field if it has been set. If the \texttt{indexcounter} field hasn’t been set (or if hyperref hasn’t been loaded) then just \texttt{$\langle text \rangle$} is done.

This provides a convenient way of encapsulating the \texttt{name} in the glossary so that it links back to the first wrglossary entry or the first \texttt{format=$\langle encap \rangle$} wrglossary entry. This encapsulation can be done by providing a new glossary style or more simply by redefining \texttt{\glsnamefont}:
Here's a complete example:
\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]

A \gls{sample}. \lipsum*[5] A \gls{duck} and \gls[format={primary}]{pi}.
\lipsum*[6] A \gls[format={primary}]{duck}.
\printunsrtglossaries
\end{document}
Note that the `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 `pi` entry.

The `format={primary}` instances indicate primary references. They’re displayed in bold (since \texttt{primary} is defined to use \texttt{hyperbf}) and these are the locations saved in the `index-counter` field because that’s the `encap` identified by the `save-index-counter={primary}` setting.

### 5.8 Supplemental Locations

*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 `record={only}` package option).

The glossaries-extra package (from v1.14) provides a way of manually adding locations in supplemental documents through the use of the `thevalue` option in the optional argument of \glsadd. Setting values manually is inconvenient and can result in errors, so bib2gls provides a way of doing this automatically. Both the main document and the supplementary document need to use the `record` option. The entries provided in the `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 `.bib` files and the same prefixes.)

For example, suppose the file `entries.bib` contains:

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

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

@abbreviation{ssi,
    short="ssi",
    long="server-side includes"
}

@index{goose,plural="geese"}
```

Now suppose the supplementary document is contained in the file `suppl.tex`:

```latex
\documentclass{article}
```

\endinput
5.8 Supplemental Locations

\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}.

\printunartglossaries
\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 approach offered by glossaries-extra is quite cumbersome and requires setting the external-location attribute and using \glsadd with the value={S1}, theHvalue={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 external-location 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 \glsxtrsupphypernumber format. Otherwise bib2gls will allow multiple sources and use the newer method.

\textbf{supplemental-locations=⟨basename⟩}

The value should be the base name (without the extension) of the supplementary document (suppl in the above example). If you have at least version 1.36 of glossaries-extra, the value
5.8 Supplemental Locations

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:

\glsxtrdisplaysupploc{\langle prefix\rangle}{\langle counter\rangle}{\langle format\rangle}{\langle src\rangle}{\langle location\rangle}

By default, this simply creates an external hyperlink to the supplementary document with the location as the hyperlink text. The hyperlink is created using \langle src\rangle as the target path with the fragment part (anchor) formed from the prefix and location. The \texttt{externallocation} attribute is not set in this case. The actual formatting is done via:

\glsxtrmultisupplocation{\langle location\rangle}{\langle src\rangle}{\langle format\rangle}

which ignores the \langle format\rangle argument by default. Its definition is simply:

\newcommand*{\glsxtrmultisupplocation}[3]{%  
  % scope required to localise changes  
  \def\glsxtrsupplocationurl{\langle src\rangle}  
  \glshypernumber{\langle location\rangle}{\langle format\rangle}  
  \glsxtrsupplocationurl{\langle src\rangle}}

This locally sets the command \texttt{\glsxtrsupplocationurl}, which is checked by \texttt{\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:

204
5.8 Supplemental Locations

\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 \glsxtrsupphypernumber, 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 \glsxtrdisplaylocnameref but the \langle file \rangle 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 fontspec is loaded) \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 locations 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:

\bibglssupplemental
{1}{\setentrycounter[1]{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:

205
If an entry has both a main location list and a supplementary location list (such as the sample entry above), the lists will be separated by \bibglssupplementalsep. The sublists (when supported) are separated by \bibglssupplementalsubsep.

**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 supplementary 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:

\begin{verbatim}
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[
    supplemental-locations={suppl},
    supplemental-selection={html,ssi},
    src={entries}]
\begin{document}
\Gls{sample} document.
\printunsrtglossaries
\end{document}
\end{verbatim}
5.9 Sorting

This will additionally add the html and ssi entries even though they haven’t been used in this document. The goose entry used in the supplementary document won’t be included.

\verb|supplemental-category=⟨value⟩|

The category field for entries containing supplemental location lists may be set using this option. If unset, ⟨value⟩ defaults to the same as that given by the category option. The ⟨value⟩ may either be a known identifier (as per category) or the category label. For example:

\begin{verbatim}
\documentclass{article}
\usepackage[hyperref]{hyperref}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[
    supplemental-locations={suppl},
    supplemental-selection={html,ssi},
    supplemental-category={supplemental},
    src={entries}]
\begin{document}
\Gls{sample} document.
\printunsrtglossaries
\end{document}
\end{verbatim}

5.9 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 5.11 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.
5.9 Sorting

For example, if a term defined with \texttt{@index} doesn’t have the \texttt{sort} field set then \texttt{bib2gls} will use the value given by the \texttt{name} field because \texttt{name} is the fallback field for \texttt{sort} for \texttt{@index} entries. If the \texttt{name} field isn’t set either then \texttt{bib2gls} will use the fallback for that field. In the case of \texttt{@index} that’s the entry’s label. If the \texttt{sort} field is explicitly set then there’s no need to use the fallback.

If, on the other hand, a term defined with \texttt{@symbol} doesn’t have the \texttt{sort} field set then \texttt{bib2gls} will use the value from the field identified by \texttt{symbol-sort-fallback}, which is the entry’s label by default (not the \texttt{name} field).

This means that if I don’t explicitly set the \texttt{sort} field for any entries then I can, for example, sort terms defined with \texttt{@index} by \texttt{name} and those defined with \texttt{@symbol} by \texttt{description} with the setting:

\texttt{symbol-sort-fallback=\{description\}}

If the field used to obtain the sort value is changed (with \texttt{sort-field}) then the \texttt{sort} field won’t be queried. This reduces the flexibility of selecting the most appropriate field for given entry types. For example, \texttt{sort-field=\{name\}} will force all entries to be sorted by the \texttt{name} field, which may not be appropriate for symbols.

If you choose a field whose value must be a label (such as \texttt{parent} or \texttt{group}) then the sort value will be that label.

You can have \texttt{@preamble} definitions that can be hidden from \texttt{bib2gls}’s interpreter. For example, \texttt{no-interpret-preamble.bib} might contain:

\texttt{@preamble{\"\providecommand{\sortop}[2]{#1 #2}\"}}

which is loaded using:

\begin{verbatim}
\GlsXtrLoadResources[src={no-interpret-preamble},
    interpret-preamble={false}]
\end{verbatim}

This provides a custom command:

\texttt{\sortop{\langle text1\rangle}{\langle text2\rangle}}

for internal use in the document. (Remember it won’t be defined on the first \LaTeX run before the .glstex file has been created and so is only used within entry fields.)

Another file, say, \texttt{interpret-preamble.bib} may provide a definition for \texttt{bib2gls}:

\texttt{@preamble{\"\providecommand{\sortop}[2]{#2, #1}\"}}

which can be processed with:

\begin{verbatim}
\GlsXtrLoadResources[src={interpret-preamble}]
\end{verbatim}

to provide \texttt{bib2gls} with this definition. The \texttt{entries.bib} file could contain:
5.9 Sorting

@entry{caesar,
name={\sortop{Gaius Julius}{Caesar}},
first={Julius Caesar},
text={Caesar},
description={Roman politician and general}
}

and then be processed with:
\GlsXtrLoadResources[src={entries}]

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 \texttt{\LaTeX} has already defined \texttt{\sortop} from the first resource set, so the name will appear as “Gaius Julius Caesar” in the glossary. (If you have \texttt{\renewcommand} rather than \texttt{\providecommand}, you can prevent the redefinition occurring in the document with \texttt{write-preamble={false}}.)

Alternatively both of these \texttt{.bib} files can be loaded in one resource set:
\GlsXtrLoadResources[src={interpret-preamble,entries}]

Another possibility is to provide a custom package that contains the command definitions for the bib2gls interpreter and load it with \texttt{--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=\{doc\}} is assumed. If the \texttt{sort} option isn’t used then \texttt{sort=\{doc\}} is assumed if the document has a language that’s been detected by \texttt{tracklang}, otherwise \texttt{sort=\{locale\}} is assumed.

The \texttt{(method)-reverse} options reverse the result returned by the corresponding \texttt{(method)} comparator. However \texttt{(method)-reverse} may not produce a list that’s the exact reverse of the underlying non-reversed \texttt{(method)} as the hierarchical structure or associated settings can affect the order.

\textbf{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 \texttt{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.)
5.9 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>doc</td>
<td>sort according to the document language</td>
</tr>
<tr>
<td>doc-reverse</td>
<td>reverse sort according to the document language</td>
</tr>
<tr>
<td>locale</td>
<td>sort according to the default locale</td>
</tr>
<tr>
<td>locale-reverse</td>
<td>reverse sort according to the default 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>
5.9 Sorting

- **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.

- **use-reverse**: reverses the order that would be obtained with `sort={use}` without reference to hierarchy.

- **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 `.glstex` 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:
5.9 Sorting

\GlsXtrLoadResources[src={entries},sort={recordcount-reverse}]

It all depends on what’s likely to be most useful to the reader. Consider the following:

\newglossary*[frequent]{Most Frequently Used Terms}
\GlsXtrLoadResources[src={entries},sort={use},
secondary={recordcount-reverse:frequent}]
\newcommand{\filterhook}[1]{%
\GlsXtrIfFieldCmpNum*{recordcount}{#1}{>}{10}%
{}%
{\printunsrtglossaryskipentry}%
}
\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},type={frequent}]{%
\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.

213
5.9 Sorting

Both examples assume there are no child entries as the filtering can cause parent entries to be omitted. Both examples require --record-count but only the first example sorts according to the record count.

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 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 27).

- (lang tag): sort according to the rules of the locale given by the IETF language tag ⟨lang tag⟩.
- (lang tag)-reverse: reverse sort according to the rules of the locale given by the IETF language tag ⟨lang tag⟩.
- locale: equivalent to sort=⟨lang tag⟩ where ⟨lang tag⟩ is obtained from the JRE’s default locale (which usually matches the operating system’s locale).
- locale-reverse: equivalent to sort=⟨lang tag⟩-reverse where ⟨lang tag⟩ is obtained from the JRE.
- doc: sort the entries according to the document language. This is equivalent to sort=⟨lang tag⟩ where ⟨lang tag⟩ is the locale associated with the document language. In the case of a multi-lingual document, ⟨lang tag⟩ 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 languages have been tracked, 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=⟨locale⟩ 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{article}  
\usepackage[ngerman]{babel}  
\usepackage[record]{glossaries}  
\GlsXtrLoadResources[src={de-1996}]  

the language tag will be de-1996.
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 language package is loaded then the default is sort={locale} rather than sort=
{doc}. 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}]

**Letter (Non Locale)**

The sort methods listed in table 5.3 use letter comparators. These simply compare the charac-
ter 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”).
5.9 Sorting

- **letter-nocase**: case-insensitive letter sort. (All upper case characters will have first been converted to lower case in the sort value.) Example:
  

- **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.
5.9 Sorting

For example, suppose the first string is \texttt{abc12foo} and the second string is \texttt{abc6bar}. Figure 5.1(a) shows the regular letter comparison using \texttt{sort=\{letter-case\}}, where the subscript indicates the hexadecimal character code. The first three characters from each string are identical (\texttt{abc}). At this point there’s no difference detected, so the comparator moves on to the next character, \texttt{131} for the first string and \texttt{636} for the second string. Since \texttt{0x31} is less than \texttt{0x36}, the first string (\texttt{abc12foo}) is considered less than the second (\texttt{abc6bar}).

With the letter-number comparison using \texttt{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 \texttt{12 > 6}, so \texttt{abc12bar} is considered greater than \texttt{abc6foo} (that is, \texttt{abc12bar} comes after \texttt{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 \texttt{१}, \texttt{२} and \texttt{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 \texttt{abc-12foo} is less than \texttt{abc+6bar}, which is again different from the result obtained with a straight letter comparator where the character + (\texttt{0x2B}) comes before the character − (\texttt{0x2D}). The sign must be followed by at least one digit.
5.9 Sorting

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 letter-number-⟨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: CH₂O, C₁₀H₁₀O₄, C₅H₄NCOOH, CO, Cl, Co, Co₂O₃, Co₂, CO₂, CoMoO₄ and CoCl₂, for the setting letter-number-rule={between}, where the subscripts are the Unicode subscript characters.

- letternumber-case: case-sensitive letter-number sort. Example:
  CH₂O, CO, CO₂, C₅H₄NCOOH, C₁₀H₁₀O₄, Cl, Co, CoCl₂, CoMoO₄, Co₂, Co₂O₃.
  (Order determined by: H < O < 5 < 10 < l < o.)

- letternumber-case-reverse: reverse case-sensitive letter-number sort. Example:
  Co₂O₃, Co₂, CoMoO₄, CoCl₂, Co, Cl, C₁₀H₁₀O₄, C₅H₄NCOOH, CO₂, CO, CH₂O.

- 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:
  C₅H₄NCOOH, C₁₀H₁₀O₄, CH₂O, Cl, Co, CO₂, CO₂, Co₂O₃, CoCl₂, CoMoO₄.
  (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:
  CoMoO₄, CoCl₂, Co₂O₃, Co₂, CO₂, Co, CO, Cl, CH₂O, C₁₀H₁₀O₄, C₅H₄NCOOH.
• **letternumber-upperlower**: upper-lower letter-number sort. This behaves slightly differently to **letternumber-upperlower** when used with **letternumber-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:

\[
\text{CH}_2\text{O}, \text{C}_5\text{H}_4\text{NCOOH}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{Cl}, \text{CO}, \text{CO}_2, \text{Co}, \text{Co}_2, \text{CoCl}_2, \text{CoMoO}_4, \text{Co}_2\text{O}_3. 
\]

(Order determined by: \( H < 5H < 10H < 1 < 0 < o \), and for the terms starting with \( 0 \) or \( Co \): 2 comes after null and \( C < M < 20 \).)

Compare this with **letternumber-rule={before letter}** which results in the order:

\[
\text{C}_5\text{H}_4\text{NCOOH}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{CH}_2\text{O}, \text{Cl}, \text{CO}, \text{CO}_2, \text{Co}, \text{Co}_2, \text{Co}_2\text{O}_3, \text{CoCl}_2, \text{CoMoO}_4. 
\]

• **letternumber-upperlower-reverse**: reverse upper-lower letter-number sort. Example (with **letternumber-rule={between}**):

\[
\text{Co}_2\text{O}_3, \text{CoMoO}_4, \text{CoCl}_2, \text{Co}_2, \text{Co}, \text{CO}_2, \text{CO}, \text{Cl}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{C}_5\text{H}_4\text{NCOOH}, \text{CH}_2\text{O}. 
\]

Compare this with **letternumber-rule={before letter}** which results in the order:

\[
\text{CoMoO}_4, \text{CoCl}_2, \text{Co}_2\text{O}_3, \text{Co}_2, \text{Co}, \text{CO}_2, \text{CO}, \text{Cl}, \text{CH}_2\text{O}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{C}_5\text{H}_4\text{NCOOH}. 
\]

Remember that the associated settings are reversed as well. So **letternumber-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 **letternumber-rule={between}** and has a more complicated rule. Example:

\[
\text{CH}_2\text{O}, \text{C}_5\text{H}_4\text{NCOOH}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{Cl}, \text{Co}, \text{Co}_2, \text{CoCl}_2, \text{CoMoO}_4, \text{Co}_2\text{O}_3, \text{CO}, \text{CO}_2. 
\]

Compare this with **letternumber-rule={before letter}** which results in the order:

\[
\text{C}_5\text{H}_4\text{NCOOH}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{CH}_2\text{O}, \text{Cl}, \text{Co}, \text{Co}_2, \text{Co}_2\text{O}_3, \text{CoCl}_2, \text{CoMoO}_4, \text{CO}, \text{CO}_2. 
\]

• **letternumber-lowerupper-reverse**: reverse lower-upper letter-number sort. Example (with **letternumber-rule={between}**):

\[
\text{CO}_2, \text{CO}, \text{Co}_2\text{O}_3, \text{CoMoO}_4, \text{CoCl}_2, \text{Co}_2, \text{Co}, \text{Cl}, \text{C}_{10}\text{H}_{10}\text{O}_4, \text{C}_5\text{H}_4\text{NCOOH}, \text{CH}_2\text{O}. 
\]

**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.
5.9 Sorting

- **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.
- **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.
5.9 Sorting

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.
- `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 \( y \times 10000 + m \times 100 + d \) 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:

```biblatex
@entry{journalentry,
    name={10 Jan 2017},
    description={an interesting journal entry}
}
```
5.9 Sorting

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:

\GlsXtrLoadResources[
   src={entries},% data in entries.bib
   sort={date},
   date-sort-locale={en-GB},
   date-sort-format={medium}
]

(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**={langle seed angle}

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**={langle field angle}

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
]

222
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. 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 ⟨field⟩, then the value of the fallback field will be used instead. If 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 sort-field={sort} setting, the fallback for the 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 entry-type-aliases) is identified in custom-sort-fallbacks, then if the sort field is missing the value is obtained from the supplied custom mapping.

- If the entry is defined using @entry (or a dual form that acts like @entry), then if the sort field is missing the value is obtained from the field identified by entry-sort-fallback. If that field is also missing then that field’s fallback is used.

- For the index entry types like @index or @indexplural, then if the sort field is missing the value is obtained from the name field. If that field is also missing, then the value is obtained from the particular entry type’s fallback for the name field. (For example, the entry’s label for @index or the plural field for @indexplural.)

- If the entry is defined with an abbreviation type (for example, @abbreviation or @acronym) then if the sort field is missing, bib2gls will fallback on the field given by abbreviation-sort-fallback.

- The symbol-like entry types fallback on the field given by symbol-sort-fallback if the sort field is missing.

- Entries defined using @bibtexentry fallback on the field given by bibtexentry-sort-fallback, which defaults to the name field. Note that this only applies to the main entry. The spawned @contributor entries behave like @index.

Use dual-sort-field when sorting dual entries.

missing-sort-fallback=⟨field⟩

With sort-field=⟨sort-field⟩, if the value of the field identified by ⟨sort-field⟩ is missing, then bib2gls behaves as follows:

1. If missing-sort-fallback=⟨fallback-field⟩ is set, then bib2gls will fallback on the value provided by the field ⟨fallback-field⟩. If ⟨fallback-field⟩ is missing, then bib2gls will query the entry type’s fallback for ⟨fallback-field⟩ (not for ⟨sort-field⟩).
The ⟨fallback-field⟩ must be a known field but not an internal field. It can’t be the sort field. (Take care not to cause an infinite loop if sort-field has been changed.) Unlike the other sort fallback options such as entry-sort-fallback, the ⟨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 ⟨sort-field⟩, then that rule is used. When sort-field={sort} this means:
   - 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.

**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:
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:

```
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`).

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:
5.9 Sorting

@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).
\printunsrtglossary
\end{document}

The default behaviour for \texttt{@entry} if the \texttt{sort} field is missing is to fallback on the \texttt{name} field. If the \texttt{name} field is missing (as with \texttt{glossarylist} and \texttt{glosscol}), then the value is obtained from the \texttt{name} field from the parent entry. The parent entry for these homographs is the \texttt{glossary} entry, which was defined with \texttt{@index} and doesn’t have the \texttt{name} field. For the \texttt{@index} entries, if \texttt{name} is missing the value is obtained from the label.

Therefore both \texttt{glossarylist} and \texttt{glosscol} end up with the same sort value: \texttt{glossary}. This triggers a message in verbose mode (\texttt{--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 \texttt{glossarylist} entry before the \texttt{glosscol} entry.

Now suppose a minor modification is made to the document:
This means that when the sort function fails to find the \texttt{sort} field for the terms defined with \texttt{@entry}, it will fallback on the \texttt{description} field. This doesn’t affect the terms defined with \texttt{@index}, which still fallback on the \texttt{name} field. This time there’s no message in the transcript file and the \texttt{glosscol} entry now comes before the \texttt{glossarylist} entry.

The \texttt{entry-sort-fallback} setting is only used when \texttt{bib2gls} tries to access the \texttt{sort} field for a term defined with \texttt{@entry} and finds that the field hasn’t been set. This means that this setting has no effect if you explicitly set the \texttt{sort} field or if you change the field used for sorting (\texttt{sort-field}).

\texttt{abbreviation-sort-fallback=\textsc{field}}

The entry types that define abbreviations (such as \texttt{@abbreviation} and \texttt{@acronym}) will, by default, fallback on the \texttt{short} field if the \texttt{sort} field is missing (assuming \texttt{sort-field=\{sort\}}). If you prefer to fallback on a different field, then you can use this option to specify the field. For example, \texttt{abbreviation-sort-fallback=\{long\}}. Note that if you use \texttt{sort-field=\{name\}}, then the fallback field will be given by \texttt{abbreviation-name-fallback} if the \texttt{name} field is omitted.

The \texttt{\textsc{field}} may be a known field but not an internal field. It can’t be the \texttt{sort} field. It may also be one of the keywords: \texttt{id} (for the entry’s label) or \texttt{original id} (for the entry’s original label). The \texttt{\textsc{field}} may also be a composite in the form \texttt{\textsc{field1}+\textsc{field2}+…} which indicates that the sort value should be obtained by concatenating the values of given fields, where the separator is given by \texttt{field-concat-sep}.

Note that \texttt{missing-sort-fallback} and \texttt{custom-sort-fallbacks} override this setting.

The \texttt{abbreviation-sort-fallback} setting is only used when \texttt{bib2gls} tries to access the \texttt{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 \texttt{sort} field or if you change the field used for sorting (\texttt{sort-field}).

\texttt{symbol-sort-fallback=\textsc{field}}

The entry types that define symbols (such as \texttt{@symbol} and \texttt{@number}) will, by default, fallback on the entry’s original label if the \texttt{sort} field is missing (assuming the default \texttt{sort-field=\{sort\}}). If you prefer to fallback on a different field, then you can use this option to specify the field. For example, \texttt{symbol-sort-fallback=\{name\}}.
5.9 Sorting

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).

field-concat-sep=⟨value⟩

This option sets the field concatenation separator to ⟨value⟩. The default is a space. An empty value indicates no separator. You may use \u⟨hex⟩ to indicate a character by its hexadecimal code. For example, suppose the .bib file contains:

@abbreviation{ac,
  short={AC},
  long={alternating current}
}
@index{acacia}

Then the resource option:
5.9 Sorting

\GlsXtrLoadResources[
  sort={letter-nocase},
  abbreviation-sort-fallback={short+long}
]

will set the \texttt{ac} sort value to “AC alternating current”. That is, the \texttt{short} value concatenated with the \texttt{long} value using the default space separator. With the \texttt{letter-nocase} sort method, this will put the \texttt{ac} entry before the \texttt{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 \texttt{short} and \texttt{long} values without a separator. This means that the \texttt{ac} sort value will be “AC alternating current” and so the \texttt{ac} entry will come after the \texttt{acacia} entry (since “I” comes after “c”).

This setting is only used for the sort fallback options that allow field concatenation (such as \texttt{entry-sort-fallback} but not \texttt{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 \texttt{field-concat-sep=} to indicate a space character as the value will end up as an empty string. You can instead do \texttt{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 \texttt{break-at} setting.

\texttt{trim-sort=}\langle boolean\rangle

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 \texttt{trim-sort=} \texttt{true}.

This option automatically sets \texttt{dual-trim-sort=}\langle boolean\rangle and \texttt{secondary-trim-sort=}\langle boolean\rangle.

\texttt{sort-replace=}\langle list\rangle

This option may be used to perform regular expression substitutions on the sort value and has the same syntax as \texttt{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 \texttt{sort-number-pad} (if applicable). For example, suppose the sort value is:

\ensuremath{\approx 3.14}
5.9 Sorting

then the interpreter will convert this to \( \approx 3.14 \) but:

\[
\text{sort-replace} = \{|\text{\glshex2248}|\}
\]

can be used to strip the \( \approx \) symbol (0x2248) so that the value can now be parsed as a number if \text{sort}={\text{double}} has been used.

Use \text{dual-sort-replace} for dual and \text{secondary-sort-replace} for secondary sort methods.

\[
\text{sort-rule} = \langle \text{value} \rangle
\]

If the \text{sort}={\text{custom}} option is used, the sort rule must be provided with \text{sort-rule}. If \text{sort} is not set to \text{custom}, the \text{sort-rule} setting will be ignored. This setting uses Java’s \text{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 .aux file, so be careful of any special characters that occur in the rule. For the special characters # % _ & \{ and \} you can use \textbackslash##, \textbackslash%\textbackslash_, \textbackslash&\textbackslash, \textbackslash{ and \textbackslash}. These will be written to the .aux file with the leading backslash, but \text{bib2gls} will remove it for this resource option. Remember that the \text{glossaries} package provides \textbackslash{\glshex} and \textbackslash{\glstildechar} which can be used to produce a literal backslash (\) and tilde (\textbackslash{}).

You can also use \textbackslash{\string\u\langle\text{hex}\rangle} (where \langle\text{hex}\rangle is a hexadecimal code) to represent a Unicode character. For example:

\[
\text{\GlsXtrLoadResources[}
\text{sort}={\text{custom}},
\text{sort-rule}=\langle a,A < b,B < c,C < ch,CH,CH < d,D}
\text{< dd,Dd,DD < e,E < f,F < ff,Ff,FF}
\text{< g,G < ng,Ng,NG < h,H < ij,Ij,IJ}
\text{< i,I < j,J < k,K < l,L < ll,Ll,LL < m,M}
\text{< n,N < o,O < p,P < ph,Ph,PH < q,Q < r,R < rh,Rh,RH}
\text{< s,S < t,T < th,Th,TH < u,U < v,V < w,W < x,X < y,Y < z,Z}
\text{< \textbackslash{\string\u00E6},\textbackslash{\string\u00C6}}
\text{]}
\]

It’s best to use \textbackslash{\string} rather than \textbackslash{\protect} to avoid unwanted spaces interfering with \langle\text{hex}\rangle. Note that \text{glossaries-extra v1.21+} provides\textbackslash{\glshex} which just does \textbackslash{\string\u} so you can do \textbackslash{\glshex 00E6} instead of \textbackslash{\string\u00E6}. This is only one character different, but you can redefine \textbackslash{\glsxtrresourceinit} to locally set \textbackslash{\u} to \textbackslash{\glshex} while the protected write is performed. For example:

\[
\text{\renewcommand*{\glsxtrresourceinit}{\let\u\glshex}}
\]

Then you can just do \textbackslash{\u00E6} instead of \textbackslash{\string\u00E6}.

\footnote{The command definition was moved to \text{glossaries-extra-bib2gls} from version 1.27 since it’s only needed with \text{bib2gls}.}
5.9 Sorting

The glossaries-extra-bib2gls package (which is automatically loaded by the \texttt{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
 ;\glsxtrspacerules
 ;\glsxtrnonprintablerules
 ;\glsxtrcombiningdiacriticrules
 ,\glsxtrhyphenrules
 <\glsxtrgeneralpuncrules
 <\glsxtrdigitrules
 <\glsxtrfractionrules
 <\glsxtrMathItalicGreekIrules
 <\glsxtrGeneralLatinIVrules
 <\glsxtrLatinAA
 <\glsxtrLatinOslash
 }
\end{verbatim}

This places the Greek maths symbols (such as \textbackslash{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}{%
 \glsxhex 2694 % crossed-swords 0x2694
 < \glsxhex 2693 % anchor 0x2693
 < \glsxhex 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}
5.9 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
  <\glsxtrgeneralpuncrules
  \glsxtrdigitrules
  \glsxtrfractionrules
  <\MapSymbolOrder % <--- custom map symbols
  <\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.

\textbf{break-at=(option)}

This option automatically implements dual-break-at=(\textit{option}) and secondary-break-at=(\textit{option}).
5.9 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 sort={letter-nocase}, which just sorts according to the Unicode value not according to a particular alphabet.)

In order to replicate makeindex and xindy’s default word order, bib2gls splits up the sort value at word boundaries and inserts a marker (identified by break-marker). For example, if the sort value is “sea lion” then it’s actually converted to sea|lion| whereas “sea” becomes sea| and “seal” becomes seal|. The default marker is | 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 parameter| (hash, space and parentheses removed). If you only want to break at spaces (optionally following a comma) use the following instead:

```latex
break-at={none},
sort-replace={{{,}? +}{|}}
```

You can change the construction of the break points with break-at={⟨option⟩} where ⟨option⟩ 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 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 MathM|L| and “W3C” becomes 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 makeindex or xindy’s letter ordering, or combine with sort-replace to insert custom break points.

This option is ignored when used with the non-alphabetic sort options. You can find the break points in the sort field for the entry’s definition in the .glstex 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.)
5.9 Sorting

**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.

**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⟩}.
5.9 Sorting

This option determines what the comparator should do if two entries at the same hierarchical level are considered equal. The \langle value \rangle 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;
- **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}.

\textbf{sort-suffix=\langle value \rangle}

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 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 sort-suffix={⟨field⟩} setting will only append a suffix if that field is set, but (if set) it will apply the suffix to all sort values, even those that are unique.

If you use --verbose, then bib2gls will write information in the transcript when it appends a suffix to the sort value. The message:

Sort value '⟨sort⟩' (entry '⟨id⟩') not unique for the entry's hierarchical level.

indicates that an entry with the given ⟨sort⟩ value has already been found within the same hierarchical level as the currently processed entry (whose label is given by ⟨id⟩). 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 '⟨suffix⟩' to the sort value '⟨sort⟩' for entry '⟨id⟩'.

which indicates that the entry (identified by the label ⟨id⟩) has been assigned the sort value given by ⟨sort⟩⟨suffix⟩. 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 makeglossaries (the application) and \makeglossaries (the command). I might decide to define semantic commands:

\newcommand*{\application}[1]{\texttt{#1}}
\newcommand*{\command}[1]{\texttt{\glsbackslash #1}}

In my .bib file I might have:

@entry{cs.makeglossaries,
   name={\command{makeglossaries}},
   category={command},
   description={opens glossary files}}
}

@entry{ap.makeglossaries,
   name={\application{makeglossaries}},
   category={application},
   description={Perl script}}
}

If bib2gls is provided with the definitions of \application and \command (by interpreting the @preamble or a package provided with --custom-packages) then it will determine that the sort value for cs.makeglossaries is \makeglossaries and the sort value for ap.makeglossaries is just makeglossaries. These are two distinct sort values from
bib2gls’s point of view although the sort rule may consider them identical if the rule ignores the \ character (such as the locale sort methods), in which case, bib2gls will then act according to identical-sort-action.

If 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 makeglossaries. The second instance will be recognised as a duplicate and the sort value will be converted to makeglossaries1 (where the automated suffix is 1 and the suffix marker, see below, is the empty string). Whereas with, say, sort-suffix-marker={.} then the sort value would become makeglossaries.1.

For comparison, consider the following document:

\documentclass{article}
\usepackage[style={indexgroup}]{glossaries}
\makeglossaries
\newcommand*{\application}{\texttt{#1}}
\newcommand*{\command}{\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:
5.9 Sorting

@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\}
\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 sort={locale} which considers \ an ignored (punctuation) character, so both \makeglossaries and makeglossaries are listed in the "M" letter group, even though the interpreter has determined that the sort value for cs.makeglossaries is the literal string \makeglossaries. Note that in this case 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 sort={letter-nocase}, then \makeglossaries will be listed in the “Symbols” letter group since the leading \ from the sort value \makeglossaries isn’t ignored with this rule.

Now let’s suppose I use interpret-preamble={false} to prevent 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 makeglossaries.

The first entry to be processed is cs.makeglossaries because it’s the first to be selected. This is assigned the sort value makeglossaries. (Note that, unless you use sort={unsrt}, the initial selection order is based on the record order. In this example, cs.makeglossaries has the first record in the .aux file.)

The next entry to be processed is ap.makeglossaries. This also ends up with the sort value makeglossaries so bib2gls converts this to makeglossaries1 and (with verbose mode on) the following messages are written to the transcript:

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'.

Both entries are listed in the “M” letter group in the order `makeglossaries, makeglossaries`
If the records are reversed:

\gls{ap.makeglossaries} and \gls{cs.makeglossaries}.

then the sort value for `cs.makeglossaries` is now considered the duplicate and the order is reversed: `makeglossaries, makeglossaries`.

Suppose now I modify the `.bib` file so that `ap.makeglossaries` is defined as:

```latex
@entry{ap.makeglossaries,
  name={\application{makeglossaries}},
  category={application},
  description={Perl script (must be used with \gls{cs.makeglossaries})}
}
```

and suppose the document only contains an explicit reference to `ap.makeglossaries`:

```latex
\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:

```latex
\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:
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.

**sort-suffix-marker**=⟨value⟩
This automatically implements the dual and secondary settings dual-sort-suffix-marker =⟨⟨value⟩⟩ and secondary-sort-suffix-marker =⟨⟨value⟩⟩.
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 sort-suffix-marker=⟨⟨value⟩⟩ where ⟨value⟩ is the marker. By default the marker is empty. You can use \string\u⟨hex⟩ or \glshex⟨hex⟩ 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 \string (for example \string#) or use the above hexadecimal markup. If you use \# it will be treated as a literal string containing a backslash followed by a hash character.

**strength**=⟨value⟩
This option automatically implements dual-strength=⟨⟨value⟩⟩ and secondary-strength =⟨⟨value⟩⟩.
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 Collator class [3] for further details.

For example, suppose the file 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:
5.9 Sorting

\documentclass{article}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[sort={en},src={entries}]
\begin{document}
gls{resume}, \gls{resumee}, \gls{RESUME}, \gls{rot}, \gls{rat}, \gls{zoo}, \gls{aardvark}.
\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 \texttt{tertiary} or \texttt{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=⟨value⟩
This option automatically implements the dual and secondary settings \texttt{dual-decomposition=⟨value⟩} and \texttt{secondary-decomposition=⟨value⟩}.
The collation decomposition used by alphabet sort methods (table 5.2) can be set to the following values: canonical (default), full or none. This determines how Unicode composed characters are handled. The fastest mode is none but is only appropriate for languages without accents. The slowest mode is 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=⟨value⟩
This automatically implements the dual and secondary settings \texttt{dual-letter-number-rule=⟨value⟩} and \texttt{secondary-letter-number-rule=⟨value⟩}.
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 ⟨value⟩ may be one of:
- \texttt{before letter}: numbers are considered less than any letter.
- \texttt{after letter}: numbers are considered greater than any letter.
• **between**: (default) numbers come between letter cases. With the *letternumber-case* sort option, this will put numbers after upper case and before lower case. This setting doesn’t make much sense with the *letternumber-nocase* option but, if used, this will put numbers before letters. The *letternumber-upperlower* and *letternumber-lowerupper* options are more complicated. See section 5.9 for more detail.

• **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);
5.9 Sorting

- **punc-last-space-zero**: punctuation comes last (although numbers may come after) and white space is replaced by the digit 0 (0x30).

- **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 \TeX{} 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:

\begin{verbatim}
C\textsubscript{10}H\textsubscript{10}O\textsubscript{4}
\end{verbatim}

will be converted to: C\textsubscript{10}H\textsubscript{10}O\textsubscript{4}, but:

\begin{verbatim}
X\textsubscript{1, 2}
\end{verbatim}

will be converted to:

\texttt{X<sub>1, 2</sub>}

which ends up as X\textsubscript{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.
5.9 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 \glshex⟨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 5.9, such as `sort={numeric}`, use this option to set the locale. The value may be:

- locale: use Java’s default locale (which is usually the operating system’s locale);
- doc: use the document’s locale or, if not set, assume `numeric-locale={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.

**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={locale}`.

The value may be `locale` (use Java’s default locale), `doc` (use the document’s 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.

**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.
5.9 Sorting

- \( \langle \text{pattern} \rangle \): provide a custom pattern. This should match the specifications for Java’s \SimpleDateFormat class \[7\]. You may use \string\u\( \langle \text{hex} \rangle \) or \gls\hex\( \langle \text{hex} \rangle \) 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 \texttt{sort={date}} or \texttt{sort={date-reverse}}. If the pattern only contains time (but not date) information, then it must be used with \texttt{sort={time}} or \texttt{sort={time-reverse}}. If the pattern contains date and time information, then it must be used with \texttt{sort={datetime}} or \texttt{sort={datetime-reverse}}.

For example, suppose each entry provides information about a person and the \texttt{user1} field is used to store their date of birth:

@entry{caesar,
   name={Gaius Julius Caesar},
   first={Julius Caesar},
   text={Caesar},
   description={Roman politician 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:

\GlsXtrLoadResources[
   src={entries},% data in entries.bib
   sort-field={user1},
   sort={date},
   date-sort-format={d MMM y G}
]

The \texttt{G} (era) date pattern specifier expects a string, such as “AD”. It will match lower case forms, such as “ad”, so if you have \texttt{\textsc{ad}} the interpreter will convert this to \texttt{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 \texttt{\era} is provided with:

\small
\begin{verbatim}
@preamble{"\providecommand{\era}[1]{\textsc{\MakeLowercase{#1}}}"}
\end{verbatim}

\normalsize
5.9 Sorting

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\bibglsdategroup[7]{#1#4#7}
\newcommand\bibglsdategrouptitle[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.)

\texttt{group-formation=\langle value\rangle}

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:

- \textbf{default}: the group is assigned according to the sort method’s default group formation. This is the default setting.

- \textbf{codepoint}: the group is set to \bibglsunicodegroup\langle label\rangle\langle character\rangle\langle id\rangle\langle type\rangle, where the first argument is the first significant character (converted to lower case and decomposed, if applicable) of the sort value.

- \textbf{unicode category}: the group is set to \bibglsunicodegroup\langle label\rangle\langle character\rangle\langle id\rangle\langle type\rangle, where the first argument is the label identifying the Unicode category.
of the first significant character of the sort value. For example, the label L1 signifies a lower case letter and Lu signifies an upper case letter.

- **unicode script**: the group is set to `\bibglsunicodetuple{⟨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 `\bibglsunicodetuple{⟨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.

## 5.10 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:

\[
\text{⟨sort⟩:⟨field⟩:⟨type⟩}
\]

or

\[
\text{⟨sort⟩:⟨type⟩}
\]

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 fallback field (such as
5.10 Secondary Glossary

(name) 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 secondarysort field in case you require it.)

The value of \langle sort \rangle 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 \texttt{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 \langle sort \rangle is custom, then the rule should be provided with secondary-sort-rule.

This option will copy all the selected entries into the glossary labelled \langle type \rangle sorted according to \langle sort \rangle (using \langle field \rangle as the sort value). Note that this just copies the entry’s label to the second 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 glossary will be stored in the internal secondarygroup field. The group field will contain the group for the primary glossary.

In order to switch fields in \texttt{\textbackslash printunsrtglossary}, you need at least v1.21 of glossaries-extra which provides \texttt{\textbackslash glxtrgroupfield} 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 secondarygroup. With older versions, the group information in the secondary glossary will be the same as for the primary glossary.

(If the glossary \langle type \rangle doesn’t exist, it will be defined with \texttt{\textbackslash provideignoredglossary* \{\langle type \}\}} even if \texttt{--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:

\texttt{@entry{quartz,} \\
\quad \texttt{name=\{quartz\},} \\
\quad \texttt{description=\{hard mineral consisting of silica\},} \\
\quad \texttt{category=\{mineral\}\}}

\texttt{@entry{cabbage,} \\
\quad \texttt{name=\{cabbage\},} \\
\quad \texttt{description=\{vegetable with thick green or purple leaves\},} \\
\quad \texttt{category=\{vegetable\}\}}

\texttt{@entry{waterfowl,} \\
\quad \texttt{name=\{waterfowl\},} \\
\quad \texttt{description=\{any bird that lives in or about water\},} \\
\quad \texttt{category=\{animal\}\}}

and the document preamble contains:
5.10 Secondary Glossary

\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},
    label-prefix={byname.}
]

\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-missing-sort-fallback=⟨field⟩
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-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=⟨value⟩
As group-formation but for secondary sorting.
5.11 Dual Entries

General Dual Settings

dual-prefix={value}

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 ⟨value⟩ when the .bib file is parsed.

As from version 1.8, the dual label prefix is identified in the .gls\text file with:

\bibglsdualprefixlabel{⟨prefix⟩}

primary-dual-dependency={boolean}

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={value}

This setting allows the location lists for each primary entry to be merged with that of the corresponding dual entry. The ⟨value⟩ 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.

For example, suppose the file entries.bib contains:

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

@dualindexentry{vector,
  name={vector},
  description={column or row of values}
}
5.11 Dual Entries

@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 \texttt{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 \texttt{gls.} and are placed in the main glossary. The \texttt{gls.array} and \texttt{gls.matrix} entries have been indexed on pages 1, 2 and 3. The \texttt{gls.vector} and \texttt{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 \texttt{gls.array}, \texttt{gls.vector}, \texttt{gls.set} and \texttt{gls.matrix}) and
some of the locations are in the index glossary (corresponding to \gls{idx.vector} and \gls{idx.set}).

If the option \texttt{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.

\section*{Dual Fields}

\texttt{dual-type=}\langle\texttt{value}\rangle

This option sets the \texttt{type} field for all dual entries. (The primary entries obey the \texttt{type} option.) This will override any value of \texttt{type} provided in the .bib file (or created through a mapping). The \langle\texttt{value}\rangle is required and should be one of:

- \texttt{same as entry}: sets the \texttt{type} to the entry type (lower case and without the initial @). For example, if the entry was defined with @dualentry, the \texttt{type} 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.

- \texttt{same as original entry}: set the \texttt{type} field to the original entry type (lower case and without the initial @) before it was aliased (behaves like \texttt{same as entry} if the entry type wasn’t aliased).

- \texttt{same as base}: sets the \texttt{type} to the base name of the .bib file (without the extension) that provided the entry definition (new to v1.1);

- \texttt{same as primary}: sets the \texttt{type} to the same as the corresponding primary entry’s \texttt{type} (which may have been set with \texttt{type}). If the primary entry doesn’t have the \texttt{type} field set, the dual’s \texttt{type} will remain unchanged.

- \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.

- \langle\texttt{label}\rangle: sets the \texttt{type} field to \langle\texttt{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}
5.11 Dual Entries

Alternatively:

\newglossary*[dictionary]{Dictionary}
\GlsXtrLoadResources[src={entries},sort={en},dual-sort={fr},
type={dictionary},
dual-type={same as primary}]
dual-category={value}

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 (value) may be empty or one of:

- **same as entry**: sets the category to the entry type (lower case and without the initial @). For example, if the entry was defined with @dualentry, the category 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 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); this may have been set with category.

- **same as primary**: sets the category to the same as the corresponding primary entry’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.

- **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.

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.
5.11 Dual Entries

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).

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.
5.11 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 sorted together according to the sort option.

If {value} isn’t set to combine then the dual entries are sorted separately according to {value} (as per sort) and the dual entries will be appended at the end of the .glstex 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, locale is assumed.
5.11 Dual Entries

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.

dual-break-marker=⟨marker⟩
As break-marker 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.
5.11 Dual Entries

dual-sort-suffix-marker=\langle value \rangle
As sort-suffix-marker but for dual entries.

dual-strength=\langle value \rangle
As strength but for dual entries.

dual-decomposition=\langle value \rangle
As decomposition but for dual entries.

dual-letter-number-rule=\langle value \rangle
As letter-number-rule but for dual entries that use a letter-number sort.

dual-letter-number-punc-rule=\langle value \rangle
As letter-number-punc-rule but for dual entries that use a letter-number sort.

dual-numeric-sort-pattern=\langle value \rangle
As numeric-sort-pattern but for dual entries that use a locale-sensitive numeric sort.

dual-numeric-locale=\langle value \rangle
As numeric-locale but for dual entries that use a locale-sensitive numeric sort.

dual-date-sort-locale=\langle value \rangle
As date-sort-locale but for dual entries that use a date/time sort.

dual-date-sort-format=\langle value \rangle
As date-sort-format but for dual entries that use a date/time sort.

dual-group-formation=\langle value \rangle
As group-formation but for dual sorting.

Dual Mappings

dual-entry-map=\{\langle list1 \rangle,\langle list2 \rangle\}
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:
5.11 Dual Entries

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 \langle list1 \rangle to the field in the corresponding place in the second list \langle list2 \rangle. Any additional fields are copied over to the same field. For example:

@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.

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, \longnewglossaryentry doesn’t provide a default for descriptionplural if it hasn’t been set.
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:

```typescript
dual-abbrv-map={
  {short, shortplural, long, longplural, dualshort, dualshortplural,
    duallong, duallongplural},
  {dualshort, dualshortplural, duallong, duallongplural, short, shortplural,
    long, longplural}
}
```

This essentially flips the `short` field with the `dualshort` field and the `long` field with the `duallong` field. See `@dualabbreviation` for further details.

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:

```typescript
dual-abbrventry-map={
  {long, short, shortplural},
  {name, text, plural}
}
```

See `@dualabbreviationentry` for further details.

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:

```typescript
dual-symbol-map={
  {name, plural, symbol, symbolplural},
  {symbol, symbolplural, name, plural}
}
```

This essentially flips the `name` field with the `symbol` field.

This is like `dual-entry-map` but applies to `@dualindexentry` rather than `@dualentry`. The default setting is:

```typescript
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.
5.11 Dual Entries

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}
}

dual-indexabbrv-map={{list1},{list2}}

This is like dual-entry-map but applies to both the dual @dualindexabbreviation and tertiary @tertiaryindexabbreviationentry entry types. The default setting is:

dual-indexabbrv-map=
  {name},
  {name}
}

**Dual Back-Links**

dual-entry-backlink={boolean}

This is a boolean setting. If boolean is missing true is assumed.

When used with @dualentry, if boolean is true, this will wrap the contents of the first mapped field with \bibglshyperlink. The field is obtained from the first mapping listed in dual-entry-map.

For example, if the document contains:

\GlsXtrLoadResources[dual-entry-backlink,
  dual-entry-map={
    {name,plural,description,descriptionplural},
    {description,descriptionplural,name,plural}
  },
  src={entries-dual}]

and if the .bib file contains:

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

Then the definition of the primary entry (child) in the .glistex file will set the description field to:
5.11 Dual Entries

\bibglshyperlink{enfant}{dual.child}

and the dual entry (\texttt{dual.child}) will have the \texttt{description} field set to:

\bibglshyperlink{child}{child}

This use of the wrapper \texttt{\bibglshyperlink} (rather than explicitly using \texttt{\glsentryname}) 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.

dual-abbrv-backlink=⟨\langle boolean⟩⟩

This is analogous to \texttt{dual-entry-backlink} but for entries defined with \texttt{@dualabbreviation} instead of \texttt{@dualentry}.

dual-symbol-backlink=⟨\langle boolean⟩⟩

This is analogous to \texttt{dual-entry-backlink} but for entries defined with \texttt{@dualsymbol} instead of \texttt{@dualentry}.

dual-abbrventry-backlink=⟨\langle 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{\glsxtdualfield} (or \texttt{\glsxtdualfield=⟨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:

\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:

\GlsXtrDualField

which defaults to \texttt{dual}. Note that if you assign a different field label with \texttt{\glsxtdualfield}, then you will need to redefine \texttt{\GlsXtrDualField} as appropriate.

For example:
5.12 Tertiary Entries

\renewcommand*{\glsuserdescription}[2]{%
  \GlsXtrDualBackLink{\glslonguserfont{#1}}{#2}%
 }
\setabbreviationstyle{long-short-user}
\GlsXtrLoadResources[src={entries},dual-field]

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}

5.12 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:
\bibglstertertiaryprefixlabel{\langle prefix\rangle}
5.12 Tertiary Entries

**tertiary-type={⟨value⟩}**

This option indicates that the tertiary entries should have their `type` field set to ⟨value⟩. If ⟨value⟩ 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.
6 Provided Commands

When \bib2gls creates the .glstex file, it writes some definitions for custom commands in the form \bibgls... which may be changed as required. The command definitions all use \providecommand which means that you can define the command with \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 \renewcommand, you will get an error on the first \LaTeX run when the .glstex file doesn’t exist. You may prefer to use \glsrenewcommand instead, which will generate a warning instead of an error.

Since many of the commands are actually used within the .glstex file, it’s best to use \newcommand before the first resource set and \renewcommand between resource sets if adjustments are necessary.

6.1 Entry Definitions

This section lists the commands (\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 .glstex file immediately before the first entry that requires it, so only the commands that are actually needed are provided.

The sort key may be set within the .glstex entry definition, but its value is usually not required in the document unless you are using a hybrid method with record=\alsoindex. After each entry is defined, if it has any associated locations and the default save-loclist =\{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 \GlsXtrSetField.

\bibglsnewentry

\begin{verbatim}
\bibglsnewentry{⟨label⟩}{⟨options⟩}{⟨name⟩}{⟨description⟩}
\end{verbatim}

This command is used to define terms identified with the @entry type. The definition provided in the .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},#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 `⟨options⟩`.

\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}]

Note that this sets the sort field to the label, but this may be overridden by the `⟨options⟩` 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 `⟨options⟩` if the entry had the category field set in the `.bib` file or if the category was overridden with `category={⟨value⟩}`.
6.1 Entry Definitions

### \bibglsnewindex

\bibglsnewindex\langle label\rangle\langle options\rangle

This command is used to define terms identified with the @index type. The definition provided in the .glstex file is:

\providecommand*{\bibglsnewindex}[2]{% 
\newglossaryentry[^{#1}]{name={#1},category={index},description={},#2}%
}

This makes the name default to the \langle label\rangle, assigns the category to index and sets an empty description. These settings may be overridden by \langle options\rangle.

Note that the description doesn’t include \nopostdesc to allow for the post-description hook used by category attributes.

### \bibglsnewindexplural

\bibglsnewindexplural\langle label\rangle\langle options\rangle\langle name\rangle

This command is used to define terms identified with the @indexplural type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewindexplural}[3]{% 
\newglossaryentry[#1]{name={#3},category={indexplural},description={},#2}%
}

This assigns the category to indexplural and sets an empty description. These settings may be overridden by \langle options\rangle.

### \bibglsnewabbreviation

\bibglsnewabbreviation\langle label\rangle\langle options\rangle\langle short\rangle\langle long\rangle

This command is used to define terms identified with the @abbreviation type. The definition provided in the .glstex file is:

\providecommand{\bibglsnewabbreviation}[4]{% 
\newabbreviation[^{#1}]{\langle label\rangle}{\langle short\rangle}{\langle long\rangle}{\langle options\rangle}%
}

Since this uses \newabbreviation, it obeys the abbreviation style for its given category (which may have been set in \langle options\rangle), either from the category field in the .bib file or through the category option). Similarly the type will obey \glsxtrabbrvtype unless the value is supplied in the .bib file or through the type option.
6.1 Entry Definitions

\bibglsnewacronym

This command is used to define terms identified with the \texttt{@acronym} type. The definition provided in the \texttt{.glstex} file is:
\begin{verbatim}
\providecommand{\bibglsnewacronym}[4]{%
  \newacronym[#2]{#1}{#3}{#4}%
}\end{verbatim}
This works in much the same way as \texttt{\bibglsnewabbreviation}. Remember that with the glossaries-extra package \texttt{\newacronym} is redefined to just use \texttt{\newabbreviation} with the default \texttt{type} set to \texttt{\acronymtype} and the default \texttt{category} set to \texttt{acronym}.

\bibglsnewdualentry

This command is used to define terms identified with the \texttt{@dualentry} type. The definition provided in the \texttt{.glstex} file is:
\begin{verbatim}
\providecommand{\bibglsnewdualentry}[4]{%
  \longnewglossaryentry*{#1}{name={#3},#2}{#4}\
}\end{verbatim}

\bibglsnewdualindexentry

This command is used to define primary terms identified with the \texttt{@dualindexentry} type. The definition provided in the \texttt{.glstex} file is:
\begin{verbatim}
\providecommand{\bibglsnewdualindexentry}[4]{%
  \longnewglossaryentry*{#1}{name={#3},category={index},#2}{}\
}\end{verbatim}
Note that this definition ignores the \texttt{(description)} argument.

\bibglsnewdualindexentrysecondary

This command is used to define secondary terms identified with the \texttt{@dualindexentry} type. The definition provided in the \texttt{.glstex} file is:
\begin{verbatim}
\providecommand{\bibglsnewdualindexentrysecondary}[4]{%
  \longnewglossaryentry*{#1}{name={#3},#2}{#4}\
}\end{verbatim}
6.1 Entry Definitions

\bibglsnewdualindexsymbol

This command is used to define primary terms identified with the \texttt{\@dualindexsymbol} type. The definition provided in the \texttt{.glstex} file is:

\providecommand{\bibglsnewdualindexsymbol}[5]{\%
\longnewglossaryentry*{\#1}{name=\#3,category=\{index\},symbol=\#4,#2}
}\%

Note that this definition ignores the \texttt{\langle description\rangle} argument.

\bibglsnewdualindexsymbolsecondary

This command is used to define secondary terms identified with the \texttt{\@dualindexsymbol} type. The definition provided in the \texttt{.glstex} file is:

\providecommand{\bibglsnewdualindexsymbolsecondary}[5]{\%
\longnewglossaryentry*{\#1}{name=\#3,category=\{symbol\},symbol=\#4,#2}
\#5\%
}

\bibglsnewdualindexnumber

This command is used to define primary terms identified with the \texttt{\@dualindexnumber} type. The definition provided in the \texttt{.glstex} file is:

\providecommand{\bibglsnewdualindexnumber}[5]{\%
\longnewglossaryentry*{\#1}{name=\#3,category=\{index\},symbol=\#4,#2}
}\%

Note that this definition ignores the \texttt{\langle description\rangle} argument.

\bibglsnewdualindexnumbersecondary

This command is used to define secondary terms identified with the \texttt{\@dualindexnumber} type. The definition provided in the \texttt{.glstex} file is:
6.1 Entry Definitions

\providecommand{\bibglsnewdualindexnumbersecondary}[5]{%
\longnewglossaryentry*{#1}{name={#3},category={number},symbol={#4},#2}
{#5}%;
}

\bibglsnewdualindexabbreviation

\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 .glistex file is:

\providecommand{\bibglsnewdualindexabbreviation}[7]{%
\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 \glsuseabbrvfont 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

\bibglsnewdualindexabbreviationsecondary{⟨label⟩}{⟨options⟩}{⟨name⟩}{⟨short⟩}{⟨long⟩}{⟨description⟩}

This command is used to define secondary terms identified with the @dualindexabbreviation entry type. The definition provided in the .glistex file is:
6.1 Entry Definitions

This ensures that a missing or empty description doesn’t interfere with the abbreviation style.

\biblsnewdualabbreviationentry

\biblsnewdualabbreviationentry\{⟨label⟩\}{⟨options⟩}{⟨short⟩}{⟨long⟩} {⟨description⟩}

This command is used to define primary terms identified with the @dualabbreviation-entry type. The definition provided in the .glstex file is:

\providecommand{\biblsnewdualabbreviationentry}[5]{%
  \newabbreviation[#2]{#1}{#3}{#4}%
}%

Note that this definition ignores the ⟨description⟩ argument.

\biblsnewdualabbreviationentrysecondary

\biblsnewdualabbreviationentrysecondary\{⟨label⟩\}{⟨options⟩}{⟨short⟩} {⟨long⟩} {⟨description⟩}

This command is used to define secondary terms identified with the @dualabbreviation-entry type. The definition provided in the .glstex file is:

\providecommand{\biblsnewdualabbreviationentrysecondary}[5]{%
  \longnewglossaryentry*{#1}{#2}{#5}%
}%

Note that this definition ignores the ⟨short⟩ and ⟨long⟩ arguments (which will typically be empty unless the default mappings are changed).

\biblsnewdualentryabbreviation

\biblsnewdualentryabbreviation\{⟨label⟩\}{⟨options⟩}{⟨short⟩}{⟨long⟩} {⟨description⟩}

This command is used to define primary terms identified with the (now deprecated) entry type @dualentryabbreviation. The definition provided in the .glstex file is:
6.1 Entry Definitions

\providecommand{\bibglsnewdualentryabbreviation}[5]{%  \newabbreviation[#2]{#1}{#3}{#4}%}

Note that this definition ignores the \textit{(description)} argument.

\bibglsnewdualentryabbreviationsecondary

\providecommand{\bibglsnewdualentryabbreviationsecondary}{\langle label\rangle\{\langle options\rangle\{\langle short\rangle\}{\langle long\rangle}\}{\langle description\rangle}}

This command is used to define secondary terms identified with the (now deprecated) entry type \texttt{@dualentryabbreviation}. The definition provided in the \texttt{.gls} file is:

\providecommand{\bibglsnewdualentryabbreviationsecondary}{\langle label\rangle\{\langle options\rangle\}{\langle long\rangle\}{\langle description\rangle}}

Note that this definition ignores the \textit{(short)} and \textit{(long)} arguments (which will typically be empty unless the default mappings are changed).

\bibglsnewdualsymbol

\providecommand{\bibglsnewdualsymbol}{\langle label\rangle\{\langle options\rangle\{\langle name\rangle\}{\langle description\rangle}}

This command is used to define terms identified with the \texttt{@dualsymbol} type. The definition provided in the \texttt{.gls} file is:

\providecommand{\bibglsnewdualsymbol}{\langle label\rangle\{\langle options\rangle\{\langle name\rangle\}{\langle description\rangle}}

\bibglsnewdualnumber

\providecommand{\bibglsnewdualnumber}{\langle label\rangle\{\langle options\rangle\{\langle name\rangle\}{\langle description\rangle}}

This command is used to define terms identified with the \texttt{@dualnumber} type. The definition provided in the \texttt{.gls} file is:

\providecommand{\bibglsnewdualnumber}{\langle label\rangle\{\langle options\rangle\{\langle name\rangle\}{\langle description\rangle}}
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}%
}\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\bibglssuseabbrvfont{#4}{\glscategory{#2}}},%  
    category={index},#3%  
  }%}
6.1 Entry Definitions

\bibglsnewtertiaryindexabbreviationentrysecondary

\bibglsnewtertiaryindexabbreviationentrysecondary{⟨label⟩}{⟨tertiary-label⟩}{⟨options⟩}{⟨tertiary-opts⟩}{⟨primary-name⟩}{⟨short⟩}{⟨long⟩}{⟨description⟩}

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}{%\newabbreviation[#3]{#1}{#6}{#7}%\longnewglossaryentry*{#2}{name={\protect\bibglsuselongfont{#7}{\glscategory{#1}}},#4}{#8}%}

The ⟨label⟩ is the label for the secondary (abbreviation) entry and ⟨tertiary-label⟩ is the label for the tertiary (regular) entry. The fifth argument (⟨primary name⟩) isn’t used but is provided if required for a custom redefinition. The name field for the tertiary is obtained from the ⟨long⟩ 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 \glsuselongfont if it’s defined otherwise it will be defined to replicate that command.

\bibglsnewbibtexentry

\bibglsnewbibtexentry{⟨label⟩}{⟨options⟩}{⟨name⟩}{⟨description⟩}

This command is used to define the main term identified with \@bibtexentry. The definition written to the .glstex file is:

\providecommand{\bibglsnewbibtexentry}{%\longnewglossaryentry*{#1}{name={#3},#2}{#4}%}

\bibglsnewcontributor

\bibglsnewcontributor{⟨label⟩}{⟨options⟩}{⟨name⟩}{⟨description⟩}

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}{%\longnewglossaryentry*{#1}{name={#3},#2}{#4}%}
6.1 Entry Definitions

\bibglsnewprogenitor

This command is used to define the main terms created by \texttt{@progenitor}. The definition is written to the .glstex file as:

\providecommand{\bibglsnewprogenitor}[4]{\% \longnewglossaryentry*{#1}{name={#3},#2}{#4}\%}

\bibglsnewspawnindex

This command is used to define the main terms created by \texttt{@spawnindex}. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawnindex}[4]{\% \longnewglossaryentry*{#1}{name={#3},#2}{#4}\%}

\bibglsnewspawnedindex

This command is used to define the terms spawned by \texttt{@progenitor} or \texttt{@spawnindex}. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawnedindex}[2]{\% \newglossaryentry{#1}{name={#1},category=index,description={} ,#2}\%}

\bibglsnewspawnindexplural

This command is used to define the main terms created by \texttt{@spawnindexplural}. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawnindexplural}[4]{\% \longnewglossaryentry*{#1}{name={#3},#2}{#4}\%}
\bibglsnewspawnedindexplural

This command is used to define the terms spawned by \@spawnindexplural. The definition is written to the \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 \@spawnentry. The definition is written to the \glstex file as:

\providecommand{\bibglsnewspawnentry}[4]{% 
\longnewglossaryentry{*}{#1}{name={#3},#2}{#4}%
}

\bibglsnewspawnedentry

This command is used to define the terms spawned by \@spawnentry. The definition is written to the \glstex file as:

\providecommand{\bibglsnewspawnedentry}[4]{% 
\longnewglossaryentry{*}{#1}{name={#3},#2}{#4}%
}

\bibglsnewspawnabbreviation

This command is used to define the main terms created by \@spawnabbreviation. The definition is written to the \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} %}
6.1 Entry Definitions

\bibglsnewspawnedsymbol

This command is used to define the terms spawned by \texttt{@spawnsymbol}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewspawnedsymbol}[4]{\%\longnewglossaryentry*{#1}{name={#3},sort={#1},category={spawnedsymbol},#2}{#4}}

\bibglsnewspawnnumber

This command is used to define the main terms created by \texttt{@spawnnumber}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewspawnnumber}[4]{\%\longnewglossaryentry*{#1}{name={#3},#2}{#4}}

\bibglsnewspawnednumber

This command is used to define the terms spawned by \texttt{@spawnnumber}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewspawnednumber}[4]{\%\longnewglossaryentry*{#1}{name={#3},sort={#1},category={spawnednumber},#2}{#4}}

\bibglsnewspawndualindexentry

This command is used to define the progenitor’s primary term created by \texttt{@spawndualindexentry}. The definition is written to the \texttt{.glistex} file as:

\providecommand{\bibglsnewspawndualindexentry}[4]{\%\longnewglossaryentry*{#1}{name={#3},category={index},#2}{#4}}

The \texttt{\langle description\rangle} argument is ignored.
6.2 Location Lists and Cross-References

This command is used to define the progenitor’s secondary (dual) term created by \texttt{@spawn-dualindexentry}. The definition is written to the .glstex file as:

\providecommand{\bibglsnewspawndualindexentrysecondary}{%
\longnewglossaryentry*{#1}{name={#3},#2}{#4}%
}

6.2 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

Any entries that provide a see field (and that field hasn’t be omitted from the location list with \texttt{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}{; }

\bibglsseealsosep

This is like \bibglsseesep but is used with cross-reference lists provided with the seealso field, if supported.
\bibglsaliassep

This is like \bibglsseesep but is used with cross-reference lists provided with the \texttt{alias} field.

\bibglsusesee

\bibglsusesee{\langle label\rangle}

Displays the formatted cross-reference list stored in the \texttt{see} field for the given entry. This just defaults to \glsxtrusesee{\langle label\rangle}.

\bibglsuseseealso

\bibglsuseseealso{\langle label\rangle}

Displays the formatted cross-reference list stored in the \texttt{seealso} field for the given entry. This just defaults to \glsxtruseseealso{\langle label\rangle}.

\bibglsusealias

\bibglsusealias{\langle label\rangle}

Displays the formatted cross-reference stored in the \texttt{alias} field for the given entry. This is defined to use \glsseeformat.

\bibglsdelimN

\bibglsdelimN

Separator between individual locations, except for the last. This defaults to \texttt{\delimN}.

\bibglslastDelimN

\bibglslastDelimN

Separator between penultimate and final individual locations. This defaults to ,~ to discourage lonely locations.
The first argument \(<\text{pattern}\>) indicates the location pattern: digit for digits, roman for lower case Roman numerals, \(\text{ROMAN}\) for upper case Roman numerals and alpha for alphabetical locations. The actual location is split into two parts, \(<\text{part1}\>) and \(<\text{part2}\>) . The string concatenation \(<\text{part1}\><\text{part2}\>) forms the actual location.

This just does \(<\text{part2}\>) by default.

If \text{max-loc-diff} is greater than 1, then any ranges that have skipped over gaps will be followed by \verb!ibglspassim!, which is defined as:

\verb!\providecommand{\bibglspassim}{ \bibglspassimname}!

You can define this before you load the \verb! .bib! file:

\verb!\newcommand{\bibglspassim}{\bibglspassimname}!\verb!\GlsXtrLoadResources[src={entries}]!

Or you can redefine it afterwards:

\verb!\GlsXtrLoadResources[src={entries}]\glsrenewcommand{\bibglspassim}{\bibglspassimname}!\verb!\bibglspassimname!

The default definition is obtained from the language resource file. For example, with \verb!bib2gls-en.xml! the provided definition is:

\verb!\providecommand{\bibglspassimname}{passim}!

Explicit ranges formed using \verb!format={()}! and \verb!format={}! or \verb!format={\langle csname\rangle}! and \verb!format={\langle csname\rangle}! (where \(<\text{csname}\>) matches and is a text-block command without the initial backslash) in the optional argument of commands like \verb!\gls! or \verb!\glsadd! are encapsulated within the argument of \verb!\bibglsrangle!. By default this simply does its argument. This command is not used with ranges that are formed by collating consecutive locations.
6.2 Location Lists and Cross-References

\bibglinterloper

\bibglinterloper\{\langle location\rangle\}

If an explicit range conflicts with a record, a warning will be issued and the conflicting record will be shifted to the front of the range inside the argument of \bibglinterloper. The default definition just does \langle location\rangle \bibglsdelimN so that it fits neatly into the list.

For example, suppose on page 4 of my document I start a range with:

\glsadd[format={()}]{sample}

and end it on page 9 with:

\glsadd[format={())}]{sample}

This forms an explicit range, but let’s suppose on page 6 I have:

\gls[format={\hyperbf}]{sample}

This record conflicts with the explicit range (which doesn’t include \hyperbf in the format). This causes a warning and the conflicting entry will be moved before the start of the explicit range resulting in 6, 4–9.

Note that implicit ranges can’t be formed from interlopers (nor can implicit ranges be merged with explicit ones), so if \gls[format={\hyperbf}]{sample} also occurs on pages 7 and 8 then the result will be 6, 7, 8, 4–9. Either remove the explicit range or remove the conflicting entries. (Alternatively, redefine \bibglinterloper to ignore its argument, which will discard the conflicting entries.)

\bibglpostlocprefix

\bibglpostlocprefix

If the \loc-prefix option is on, \bibglpostlocprefix will be inserted at the start of location lists, and its default definition includes \bibglpostlocprefix placed after the prefix text. This command is provided with:

\providecommand{\bibglpostlocprefix}{\hspace{}}

which puts a space between the prefix text and the location list. You can define this before you load the .bib file:

\newcommand{\bibglpostlocprefix}{\hspace{}}
\GlsXtrLoadResources[src={entries},loc-prefix]

Or you can redefine it afterwards:

\GlsXtrLoadResources[src={entries},loc-prefix]
\glsrenewcommand{\bibglpostlocprefix}{\hspace{}}
6.2 Location Lists and Cross-References

\bibglslocprefix

\bibglslocprefix{⟨n⟩}

If the \texttt{loc-prefix} option is on, this command will be provided. If the glossary type has been provided by \texttt{type} (and \texttt{dual-type} if there are any dual entries) then the definition of \bibglslocprefix will be appended to the glossary preamble for the given type (or types if there are dual entries). For example, if the document has:

\GlsXtrLoadResources[type={main},loc-prefix={p.,pp.},src={entries}]

and there are no dual entries, then the following will be added to the \texttt{.glstex} file:

\apptoglossarypreamble[main]{%
\providecommand{\bibglslocprefix}[1]{%
\ifcase##1
\or p.\bibglspostlocprefix
\else pp.\bibglspostlocprefix
\fi
}%
}%

However, if the \texttt{type} key is missing, then the following will be added instead:

\appto\glossarypreamble{%
\providecommand{\bibglslocprefix}[1]{%
\ifcase#1
\or p.\bibglspostlocprefix
\else pp.\bibglspostlocprefix
\fi
}%
}%

\bibglspagename

\bibglspagename

If \texttt{loc-prefix={true}} is used, then this command is provided using the value of \texttt{tag.page} from the language resource file. For example with \texttt{bib2gls-en.xml} the definition is:

\providecommand{\bibglspagename}{Page}
If \texttt{loc-prefix={true}} is used, then this command is provided using the value of \texttt{tag.pages} from the language resource file. For example with \texttt{bib2gls-en.xml} the definition is:

\providecommand{\bibglspagesname}{Pages}

\bibglslocsuffix{\langle n\rangle}

If the \texttt{loc-suffix} option is on, this command will be provided. If the glossary type has been provided by \texttt{type} (and \texttt{dual-type} if there are any dual entries) then the definition of \texttt{\bibglslocsuffix} will be appended to the glossary preamble for the given type (or types if there are dual entries).

This command’s definition depends on the value provided by \texttt{loc-suffix}. For example, with \texttt{loc-suffix={\@.}} the command is defined as:

\providecommand{\bibglslocsuffix}[1]{\@.}

(which ignores the argument).

Whereas with \texttt{loc-suffix={\langle A\rangle,\langle B\rangle,\langle C\rangle}} the command is defined as:

\providecommand{\bibglslocsuffix}[1]{\ifcase#1 \langle A\rangle \or \langle B\rangle \else \langle C\rangle \fi}

Note that this is slightly different from \texttt{\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.

\bibglspprimary{\langle n\rangle}{\langle locations\rangle}

When the \texttt{save-primary-locations} option is used, the primary locations are stored in the \texttt{primarylocations} field encapsulated with this command. 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.

\bibglslocationgroup{\langle n\rangle}{\langle counter\rangle}{\langle list\rangle}

When the \texttt{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 \texttt{loc-counters}). Each group
of locations is encapsulated within \bibglslocationgroup, where \( n \) is the number of locations within the group, \( \langle \text{counter} \rangle \) is the counter name and \( \langle \text{list} \rangle \) is the formatted location sub-list. By default, this simply does \( \langle \text{list} \rangle \), but may be defined (before the resources are loaded) or redefined (after the resources are loaded) as required.

For example:

\begin{verbatim}
\newcommand*{\bibglslocationgroup}[3]{% 
  \ifnum#1=1
  #2:
  \else
  #2s:
  \fi
  #3%
}\end{verbatim}

\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:

\begin{verbatim}
\providecommand{\pagename}{Page} 
\providecommand{\pagesname}{Pages} 
\providecommand{\equationname}{Equation} 
\providecommand{\equationsname}{Equations} 
\newcommand*{\bibglslocationgroup}[3]{% 
  \ifnum#1=1 
  \ifsdef{#2name}{\csuse{#2name}}{#2}:
  \else
  \ifsdef{#2sname}{\csuse{#2sname}}{#2s}:
  \fi
  #3%
  \fi
}\end{verbatim}

\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:
6.2 Location Lists and Cross-References

\newcommand*{\bibglslocationgroupsep}{; }

\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}{; }

\bibglssupplemental

\bibglssupplemental{(n)}{(list)}

When the supplemental-locations option is used, the locations from a supplementary
document are encapsulated within the \langle list \rangle part of \bibglssupplemental. The first argu-
ment \langle n \rangle (ignored by default) is the number of supplementary locations.

If multiple supplemental sources are permitted (that is, bib2gls has detected that the
document is using at least version 1.36 of glossaries-extra), then the \langle 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 external document \rangle
(ignored by default) is the external source and \langle list \rangle is the list of supplementary locations in
\langle external document \rangle.

\bibglssupplementalsep

\bibglssupplementalsep

The separator between the main location list and the supplementary location list. By default
this is just \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}{; }

\bibglssupplementalsubsep

The separator between the supplementary location sub-lists. By default this is just \bibglsdelimN.

\bibglshrefchar

\bibglshrefchar{\langle hex\rangle}{\langle char\rangle}

Expands to a literal percent character followed by \langle hex\rangle. The second argument is ignored.

\bibglshrefunicode

\bibglshrefunicode{\langle hex\rangle}{\langle char\rangle}

Expands to the second argument. The first argument is ignored.

\bibglshexunicodechar

\bibglshexunicodechar{\langle hex\rangle}

This command is used by the hex-unicode-fields option when replacing any Unicode characters. The argument \langle hex\rangle 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{"\langle hex\rangle} (which will require an appropriate font in order to render the symbol correctly).

289
6.3 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:

\texttt{\glsgroupheading{⟨label⟩}}

to the associated glossary file at the start of each new letter group. For example, the “A” letter group will be written as:

\texttt{\glsgroupheading{A}}

This is quite straightforward and the heading title can just be “A”. The “Symbols” group is written as:

\texttt{\glsgroupheading{glssymbols}}

To allow for easy translation, the base glossaries package has the simple rule:

- if \texttt{⟨heading⟩}groupname exists use that;
- otherwise just use \texttt{⟨heading⟩}.

There’s no \texttt{\Agroupname} provided, but \texttt{\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 \texttt{⟨heading⟩} to form the target name. A problem arises when active characters occur in \texttt{⟨heading⟩}, which happens with extended characters and inputenc.

The glossaries-extra package (as from version 1.14) provides:

\texttt{\glsxtrsetgrouptitle{⟨group label⟩}{⟨group title⟩}}

to set the title for a group with the given label. The internal workings of \texttt{\glsgroupheading} are modified to use a slightly altered rule:

- if a title has been set using \texttt{\glsxtrsetgrouptitle{⟨heading⟩}{⟨title⟩}} for the given \texttt{⟨heading⟩}, use that;
- if \texttt{⟨heading⟩}groupname exists, use that;
- just use \texttt{⟨heading⟩} for the title.
6.3 Letter Groups

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 .glistex 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 \bibglsunicodergroup{{\textlangle title\textrangle}}{{\textlangle letter\textrangle}}{{\textlangle id\textrangle}}{{\textlangle type\textrangle}} and the title is set to \bibglsunicodergrouptitle{{\textlangle label\textrangle}}{{\textlangle character\textrangle}}{{\textlangle id\textrangle}}{{\textlangle type\textrangle}} (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):

- \textlangle title\textrangle 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.\langle lc\rangle exists, where \langle lc\rangle is the lower case version of \langle title\rangle, 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:

  \begin{quote}
  \textltentry key="grouptitle.case.ij">IJ</entry>
  \end{quote}

  (See the --group switch for more details.)

- \textlangle letter\textrangle 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 \textlangle title\textrangle.

- \textlangle id\textrangle A numeric identifier. This may be the collation key or the code point for the given letter, depending on the sort method.

- \textlangle type\textrangle The entry’s glossary type. If not known, this will be empty. (bib2gls won’t know if you’ve modified the associated \textbackslash 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:

\texttt{group=\textbackslash bibglslettergroup{{\textlangle title\textrangle}}{{\textlangle letter\textrangle}}{{\textlangle id\textrangle}}{{\textlangle type\textrangle}}}}

This field needs to expand to a simple label, which \textbackslash bibglslettergroup is designed to do. Note that non-letter groups are dealt with separately (see below).
6.3 Letter Groups

\bibglsetlastgrouptitle

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:

\bibglsetlastgrouptitle{⟨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 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 ⟨cs⟩ argument is the control sequence used in the group field to obtain the label from ⟨specs⟩. For example, if the highest ⟨id⟩ is 2147418112 from:

\begin{verbatim}
group={\bibglslettergroup{Ø}{Ø}{2147418112}{}}
\end{verbatim}

then the last group is identified with:

\bibglsetlastgrouptitle{\bibglslettergroup{{Ø}{Ø}{2147418112}{}}}

In this case ⟨cs⟩ is \bibglslettergroup and ⟨specs⟩ are the arguments for that command. If you want \bibglsetlastgrouptitle to change the group title then you need to define it before the resource set. For example:

\begin{verbatim}
newcommand*{\bibglsetlastgrouptitle}[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
{\bibglslettergroup{{Ø}{Ø}{2147418112}{}}}% label
{Foreign Words} % title
\end{verbatim}

\bibglsetlettergrouptitle

For each letter group that’s detected, bib2gls will write the line:

\bibglsetlettergrouptitle{⟨⟨title⟩⟩}{⟨⟨letter⟩⟩}{⟨⟨id⟩⟩}{⟨⟨type⟩⟩}

in the .glstex file, which sets the group’s title using:

\glsxtrsetgrouptitle{⟨⟨group label⟩⟩}{⟨⟨group title⟩⟩}
where the \textit{group label} part matches the corresponding \textit{group} value.

Note that \texttt{\bibglsetlettergrouptitle} only has a single argument, but that argument contains the four arguments needed by \texttt{\bibglsetlettergroup} and \texttt{\bibglsetlettergrouptitle}. These arguments are as described above.

If \texttt{\glsxtrsetgrouptitle} has been defined (glossaries-extra version 1.14 onwards), then \texttt{\bibglsetlettergrouptitle} will be defined as:

\begin{verbatim}
\providecommand{\bibglsetlettergrouptitle}[1]{%
  \glsxtrsetgrouptitle{\bibglsetlettergroup#1}{\bibglsetlettergrouptitle#1}}
\end{verbatim}

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 \textit{title} as the label.

Since \texttt{\bibglsetlettergrouptitle} is used in the \texttt{.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 \texttt{.glstex} file for the appropriate arguments. For example, if the \texttt{.glstex} file contains:

\begin{verbatim}
\bibglsetlettergrouptitle{\{E\}{æ}{7274496}{}}
\end{verbatim}

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:

\begin{verbatim}
\glsxtrsetgrouptitle{
  \bibglsetlettergroup{E}{æ}{7274496}{}}% label
  {Æ (AE)}% title
\end{verbatim}

\texttt{\bibglsetlettergroup}

\begin{verbatim}
\bibglsetlettergroup{(title)}{(letter)}{(id)}{(type)}
\end{verbatim}

This command is used to determine the letter group label. The default definition is \texttt{\(\langle\text{type}\rangle\langle\text{id}\rangle\)}, which ensures that no problematic characters occur in the label since \texttt{\(\langle\text{type}\rangle\)} can't contain special characters and \texttt{\(\langle\text{id}\rangle\)} is numeric. The \texttt{\(\langle\text{type}\rangle\)} is included in case there are multiple glossaries, since the hyperlink name must be unique.

\texttt{\bibglsetlettergrouptitle}

\begin{verbatim}
\bibglsetlettergrouptitle{(title)}{(letter)}{(id)}{(type)}
\end{verbatim}

This command is used to determine the letter group title. The default definition is \texttt{\unexpanded{(title)}}, which guards against any expansion issues that may arise with characters outside the basic Latin set.

For example:
6.3 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 secondary entries
    type={japanese},% put the primary entries in the 'japanese' glossary
    dual-type={english},% put the dual entries 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}}}

294
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:

\newcommand*{\englishlettergroup}{1}{1}
\newcommand*{\japaneselettergroup}{1}{\cjkname{1}}
\newcommand{\bibgslettergrouptitle}{4}{\%
\unexpanded{\csuse{4lettergroup}{1}}}

\texttt{\bibglssetothergrouptitle}

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:

\texttt{\bibglssetothergrouptitle{⟨character⟩}{⟨id⟩}{⟨type⟩}}

This is defined in an analogous manner:

\providecommand{\bibglssetothergrouptitle}{1}{%
\glsxtrsetgrouptitle{\bibglsothergroup#1}{\bibglsothergrouptitle#1}}

where the group label is obtained using \texttt{\bibglsothergroup} and the group title is obtained from \texttt{\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 sort={letter-case}, then this entry will be identified as belonging to a symbol (or “other”) group and the title will be set using:

\bibglssetothergrouptitle{/{}/2F{}}

\texttt{\bibglsothergroup{⟨character⟩}{⟨id⟩}{⟨type⟩}}

This expands to the label for symbol groups. This just defaults to \texttt{glssymbols} (ignoring all arguments), which replicates the label used when makeindex or xindy generate the glossary files.
6.3 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 \bibgl...grouptitle commands.)

\bibglemptygroup

\bibglemptygroup\{\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).

\bibglemptygrouptitle

\bibglemptygrouptitle\{\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{\bibglstrlenumbergroup#1}{\bibglsetnumbergrouptitle#1}}

296
\bibglsnumbergroup

The number group label is obtained from:

\bibglsnumbergroup{⟨value⟩}{⟨id⟩}{⟨type⟩}

This just defaults to \glsnumbers.

\bibglsnumbergrouptitle

The number group title is obtained from:

\bibglsnumbergrouptitle{⟨value⟩}{⟨id⟩}{⟨type⟩}

This just defaults to \glsnumbersgroupname.

\bibglssetdatetimegrouptitle

The date-time sort methods (table 5.6) create date-time groups. These behave in an analogous way to the above.

\bibglssetdatetimegrouptitle{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}{⟨hh⟩}{⟨mm⟩}{⟨ss⟩}
{⟨zone⟩}{⟨title⟩}{⟨group-id⟩}{⟨type⟩}

This command is defined as:

\providecommand{\bibglssetdatetimegrouptitle}[1]{% \
\glsxtrsetgrouptitle \\
{\bibglsdatetimegroup\hbox{#1}}% \\
{\bibglsdatetimegrouptitle\hbox{#1}}% 
}%

\bibglsdatetimegroup

This command is used for date-time group labels with 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{\bibglsdatetimegroup}[9]{\@firstofone}

This forms the group label from the year ⟨YYYY⟩, month ⟨MM⟩, day ⟨DD⟩ and ⟨type⟩.
6.3 Letter Groups

\bibglsdatetimengrouptitle

\bibglsdatetimengrouptitle\{\langle YYYY \rangle\} \{\langle MM \rangle\} \{\langle DD \rangle\} \{\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 used for date-time group titles with datetime sorting (table 5.6). The default definition is:

\providecommand{\bibglsdatetimengrouptitle}\[9\]{#1-#2-#3@gobble}

This sets the title to the numeric \(\langle YYYY \rangle-\langle MM \rangle-\langle DD \rangle\) but may be redefined as appropriate.

\bibglssetdategrouptitle

The date sort methods (table 5.6) create date groups (the time isn’t included). These behave in an analogous way to the above.

\bibglssetdategrouptitle\{\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{\bibglssetdategrouptitle}\[1\]\{%
\glsxtrsetgrouptitle{\bibglsdategroup\#1}{\bibglsdategrouptitle\#1}\}

\bibglsdategroup

\bibglsdategroup\{\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{\bibglsdategroup}\[7\]\{#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.

\bibglsdategrouptitle

\bibglsdategrouptitle\{\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{\bibglsdategrouptitle}\[7\]\{#1-#2\}

This just sets the title to the numeric year-month form \(\langle YYYY \rangle-\langle MM \rangle\).
\bibglsettimegrouptitle
The time sort methods (table 5.6) create time groups (the date isn’t included). These behave in an analogous way to the above.

\bibglsettimegrouptitle{\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{\bibglsettimegrouptitle}[1]{{\%\glsxtrsetgrouptitle\bibglstimegrouptitle#1}\bibglstimegrouptitle#1}}

\bibglstimegroup
\bibglstimegroup{\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 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
\bibglstimegrouptitle{\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 used for time group titles with time (no date) sorting (table 5.6). This command is defined as:
\providecommand{\bibglstimegrouptitle}[7]{#1#1}

\bibglsetunisegroup
\bibglsetunisegroup{\langle label\rangle\langle character\rangle\langle id\rangle\langle type\rangle}

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 \bibglsetunisegroup and the title is obtained from \bibglsetunisegrouptitle.

\bibglsetunisegroup
\bibglsetunisegroup{\langle label\rangle\langle character\rangle\langle id\rangle\langle type\rangle}

The \langle label\rangle depends on the group-formation setting:
6.3 Letter Groups

- `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 `<type><label>`.

The `<character>` is the first significant character of the sort value. The `<id>` is the hexadecimal code of (possibly decomposed) `<character>`. The case of codepoint `<id>` may or may not correspond to the case of `<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:

\[
\text{group=}\{\text{\verb|\bibglsunicodegroup|\{å\}{Å}\{C5\}}\}
\]

whereas with `group-formation={unicode category and script}` it will be:

\[
\text{group=}\{\text{\verb|\bibglsunicodegroup|\{Lu.LATIN\}{Å}\{C5\}}\}
\]

(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:

\[
\text{group=}\{\text{\verb|\bibglsunicodegroup|\{a\}{Å}\{61\}}\}
\]

Note that the `<id>` is now 0x61 (the decomposed “A” converted to lower case) not 0xC5.

\bibglsunicodegrouptitle

The title for Unicode group formations is simply defined as \verb|\unexpanded{\langle label\rangle}| so you will need to change it to something more appropriate. For example (before the resource set):

\newcommand{\bibglsunicodegrouptitle}[4]{%
  \ifnum"#3>64
    \ifnum"#3 < 91
      \verb|A--Z|%
    \else
      \verb|\quad|%
    \fi
  \fi
  \else
    \verb|\quad|%
  \fi
}
6.4 Flattened Entries

This will make the title “A–Z” if \( \langle id \rangle \) 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:

\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.

\bibglshypergroup

\bibglshypergroup{\langle type \rangle}{\langle group-id \rangle}

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.

6.4 Flattened Entries

These commands relate to the way the name field is altered when flattening lonely child entries with the flatten-lonely option.

\bibglshypergroup

\bibglshypergroup{\langle name \rangle}{\langle parent label \rangle}

The default definition simply does \langle name \rangle.

This command is used if the child and parent name’s are identical. For example, suppose the .bib file contains:

@index{super.glossary, name={glossary}}

@entry{glossarycol,
6.4 Flattened Entries

```latex
\documentclass{article}
\usepackage[record,subentrycounter,style={treenoname}]{glossaries-extra}
\GlsXtrLoadResources[src={entries}]
\begin{document}
\printunsrtglossary
\end{document}
```

This uses one of the glossary styles designed for homographs and the glossary has the structure:

**glossary**

1) collection of glosses
2) list of technical words

If only one child entry is selected, then the result looks a little odd. For example:

**glossary**

1) collection of glosses

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 `<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:

\bibglsflattenedhomograph{glossary}{super.glossary}

(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.
6.5 Other

\bibglsflattenedchildpresort

\bibglsflattenedchildpresort\{(child name)\}{(parent name)}

Used by the flatten-lonely={presort} option. This defaults to just (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)}"}

\bibglsflattenedchildpostsort

\bibglsflattenedchildpostsort\{(parent name)\}{(child name)}

Used by the flatten-lonely={postsort} option. This defaults to (parent name), (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).

6.5 Other

\bibglshyperlink

\bibglshyperlink\{(text)\}{(label)}

Used by the back link options, this just defaults to:

\glshyperlink\{(text)\}{(label)}

\bibglssetwidest

\bibglssetwidest\{(level)\}{(name)}

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:

\providecommand*{\bibglssetwidest}[2]{\glsxtrSetWidest{}{#1}{#2}}

if \glsxtrSetWidest has been defined, or:
6.5 Other

\providecommand*{\bibglssetwidest}[2]{\glsupdatewidest[#1]{#2}}

if \glsupdatewidest is defined, otherwise it will be defined to use \glssetwidest:

\providecommand*{\bibglssetwidest}[2]{\glssetwidest[#1]{#2}}

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.

\bibglssetwidestfortype

\bibglssetwidestfortype{⟨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*{\bibglssetwidestfortype}[3]{%  \glsxtrSetWidest{#1}{#2}{#3}%;%
%
}

if \glsxtrSetWidest has been defined, or:

\providecommand*{\bibglssetwidestfortype}[3]{%  \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%;%
%
}

if \glsupdatewidest is defined, otherwise it will be defined to use \glssetwidest:

\providecommand*{\bibglssetwidestfortype}[3]{%  \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%;%
%
}

Since the glossary preamble is scoped, this won’t affect other glossaries.

\bibglssetwidestfallback

\bibglssetwidestfallback{⟨glossary list⟩}

This is used by set-widest 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{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 \texttt{set-widest} instead of \texttt{\bibglsetwidestfortype} when all \texttt{name} fields end up as an empty string when interpreted by \texttt{bib2gls}. This typically means that all the \texttt{name} fields contain unknown commands. This fallback command will append \texttt{\bibglset-widestfallback} to the glossary preamble for the given type.

\bibglsetwidesttoplevelfallback

\bibglsetwidesttoplevelfallback{(glossary list)}

This is used by \texttt{set-widest} instead of \texttt{\bibglsetwidest} when all \texttt{name} fields end up as an empty string when interpreted by \texttt{bib2gls}. This typically means that all the \texttt{name} fields contain unknown commands. This fallback command will use:

\texttt{\glsxtrSetWidestFallback{0}{(glossary list)}}

if defined otherwise it will use \texttt{\glslsFindWidestTopLevelName}, which sets the widest name for the top-level.

\bibglsetwidesttoplevelfortypefallback

\bibglsetwidesttoplevelfortypefallback{(type)}

This is used by \texttt{set-widest} instead of \texttt{\bibglsetwidestfortype} when all \texttt{name} fields end up as an empty string when interpreted by \texttt{bib2gls}. This typically means that all the \texttt{name} fields contain unknown commands. This fallback command will append \texttt{\bibglset-widesttoplevelfallback} to the glossary preamble of the given type.

\bibglsccontributorlist

\bibglsccontributorlist{(list)}{(number)}

This is used when \texttt{bibtex-con contributor-fields} is set. The definition depends on whether or not \texttt{\DTLformatlist} has been defined:

\ifdef\DTLformatlist
{% datatool v2.28+
 \providecommand*{\bibglsccontributorlist}[2]{\DTLformatlist[#1]}
} {%

ifdef\DTLformatlist
{% datatool v2.28+
 \providecommand*{\bibglsccontributorlist}[2]{\DTLformatlist[#1]}
} {%


305
6.5 Other

\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
name:
\else
names:
\fi
\DTLformatlist{#1}
}
\bibglscontributor

\bibglscontributor{⟨forenames⟩}{⟨von-part⟩}{⟨surname⟩}{⟨suffix⟩}

This is used when \bibtex-contributor-fields is set. The definition depends on the value of \contributor-order. Note that if you have multiple resource sets, that option governs the way \bib2gls’s version of \bibglscontributor behaves. The definition is written to the \glstex using \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 \date-time-fields. Since \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}
6.5 Other

\bibglsdate

\bibglsdate{(year)}{(month)}{(day-of-month)}{(day-of-week)}{(day-of-year)}{(era)}{(original)}

Used to encapsulate date fields identified with date-fields. The default definition is:

\providecommand{\bibglsdate}{#7}

\bibglstime

\bibglstime{(hour)}{(minute)}{(second)}{(millsec)}{(dst)}{(zone)}{(original)}

Used to encapsulate date fields identified with time-fields. The default definition is:

\providecommand{\bibglstime}{#7}

\bibglsprimaryprefixlabel

\bibglsprimaryprefixlabel{(prefix)}

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.

\bibglsprimaryprefixlabel

\bibglsprimaryprefixlabel{(prefix)}

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.

\bibglstertiaryprefixlabel

\bibglstertiaryprefixlabel{(prefix)}

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.
\bibglsexternalprefixlabel

\bibglsexternalprefixlabel\{\langle n\rangle \langle \text{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.

\bibglshashchar

\bibglshashchar

Expands to a literal hash character (\#).

\bibglsunderscorechar

\bibglsunderscorechar

Expands to a literal underscore character (_).

\bibglsdollarchar

\bibglsdollarchar

Expands to a literal dollar character ($).

\bibglsampersandchar

\bibglsampersandchar

Expands to a literal ampersand character (&).

\bibglscircumchar

\bibglscircumchar

Expands to a literal circumflex character (^).

\bibglssupercase

\bibglssupercase\{\langle \text{text} \rangle\}

Converts \langle \text{text} \rangle to upper case. This just uses \MakeTextUppercase by default.
\bibglslowercase

\bibglslowercase{⟨text⟩}

Converts ⟨text⟩ to lower case. This just uses \MakeTextLowercase by default.

\bibglstitlecase

\bibglstitlecase{⟨text⟩}

Converts ⟨text⟩ to title case. This just uses \capitalisewords by default.

\bibglsfirstuc

\bibglsfirstuc{⟨text⟩}

Converts the first letter of ⟨text⟩ to upper case. This just uses \makefirstuc by default.
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
```
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 @*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-category 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 *type*.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).
--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 \glsxtrnoppunc. (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 overwritten. (Default unless --split-on-type.)

--no-overwrite
Don’t allow existing .bib files to be overwritten. (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 \texttt{⟨key=value list⟩} contains:

\texttt{see=\{[\seesymbolname](\texttt{label(s)})\}}

or

\texttt{see=\{[\alsymbolname](\texttt{label(s)})\}}

do not appear in the .tex file; this will be substituted with:
7.2 Recognised Commands

\seealso{⟨label(s)⟩}

For example:
\newterm[see={\seealso{goose}}]{duck}

will be written as:
@index{duck,
    seealso = {goose}
}

Note that it won’t convert see={\seealso⟨labels⟩}. If you have used explicit text instead of \seealso{} or \seealso{} then consider performing a global search and replace on your file using your text editor.

Additionally, if ⟨key=value list⟩ contains:
\type={\glsdefaulttype}

then this field will be ignored. (This type value is recommended in ⟨key=value list⟩ when loading files with \loadglsentries[⟨type⟩]{⟨file⟩} 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 \LaTeX{} (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{⟨field⟩}

If present, this instructs convertgls2bib to expand the given field, even if \glsnoexpandfields has been used.
7.2 Recognised Commands

\texttt{\textbackslash glssetnoexpandfield}

The base glossaries package provides:

\begin{lstlisting}[language=latex]
\texttt{\textbackslash glssetnoexpandfield\{\textit{field}\}}
\end{lstlisting}

If present, this instructs convertgls2bib to not expand the given field, even if \texttt{\textbackslash glsexpandfields} has been used. Unlike the default behaviour with the glossaries package, there are no fields switched explicitly switched off by default with convertgls2bib.

\texttt{\textbackslash newglossaryentry}

The base glossaries package provides:

\begin{lstlisting}[language=latex]
\texttt{\textbackslash newglossaryentry\{\textit{label}\}\{\textit{key}=\textit{value list}\}}
\end{lstlisting}

This is converted to:

\begin{lstlisting}[language=latex]
\texttt{@entry\{\textit{label},
  \hspace{1em}\textit{key}=\textit{value list}
\}}
\end{lstlisting}

\texttt{\textbackslash newentry} is recognised as a synonym of \texttt{\textbackslash newglossaryentry}.

\texttt{\textbackslash provideglossaryentry}

The base glossaries package provides:

\begin{lstlisting}[language=latex]
\texttt{\textbackslash provideglossaryentry\{\textit{label}\}\{\textit{key}=\textit{value list}\}}
\end{lstlisting}

This is converted to:

\begin{lstlisting}[language=latex]
\texttt{@entry\{\textit{label},
  \hspace{1em}\textit{key}=\textit{value list}
\}}
\end{lstlisting}

but only if \texttt{\textit{label}} hasn’t already been defined.

\texttt{\textbackslash longnewglossaryentry}

The base glossaries package provides:

\begin{lstlisting}[language=latex]
\texttt{\textbackslash longnewglossaryentry\{\textit{label}\}\{\textit{key}=\textit{value list}\}\{\textit{description}\}}
\end{lstlisting}

This is converted to:

\begin{lstlisting}[language=latex]
\texttt{@entry\{\textit{label},
  \hspace{1em}\textit{key}=\textit{value list},
  \hspace{1em}description = \{\textit{description}\}
\}}
\end{lstlisting}
7.2 Recognised Commands

The starred version provided by the glossaries-extra package is also recognised. The un-starred version strips trailing spaces from \(\langle\text{description}\rangle\). (This doesn’t add \texttt{\noindent}, but glossaries-extra defaults to \texttt{nopostdot}.)

\texttt{\longprovideglossaryentry}

The base glossaries package provides:

\begin{verbatim}
\longprovideglossaryentry{\langle\text{label}\rangle}{\langle\text{key}=\text{value list}\rangle}{\langle\text{description}\rangle}
\end{verbatim}

As above, but only if \(\langle\text{label}\rangle\) hasn’t already been defined.

\texttt{\newterm}

The base glossaries package provides:

\begin{verbatim}
\newterm[\langle\text{key}=\text{value list}\rangle]{\langle\text{label}\rangle}
\end{verbatim}

(when the \texttt{index} option is used). This is converted to:

\begin{verbatim}
@index{\langle\text{label}\rangle},
  \langle\text{key}=\text{value list}\rangle
}\n\end{verbatim}

if the optional argument is present, otherwise it’s just converted to:

\begin{verbatim}
@index{\langle\text{label}\rangle}
\end{verbatim}

If \texttt{--space-sub} is used and \(\langle\text{label}\rangle\) contains one or more spaces, then \texttt{name} will be set if not included in \(\langle\text{key}=\text{value list}\rangle\). For example, if \texttt{entries.bib} contains:

\begin{verbatim}
\newterm{sea lion}
\newterm[seealso={sea lion}]{seal}
\end{verbatim}

then:

\texttt{convertgls2bib --space-sub \'-\' entries.bib entries.tex}

will write the terms to \texttt{entries.tex} as:

\begin{verbatim}
@index{sea-lion,
   name = {sea lion}
}
@index{seal,
  seealso = {sea-lion}
}
\end{verbatim}

whereas just:
7.2 Recognised Commands

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).

\newabbreviation

The glossaries-extra package provides:

\newabbreviation\[\langle key=value list \rangle\{\langle label \rangle\}{\langle short \rangle}{\langle long \rangle}\]

This is converted to:

@abbreviation\{\langle label \rangle, 
short = {\langle short \rangle}, 
long = {\langle long \rangle}, 
\langle key=value list \rangle\}

if the optional argument is present, otherwise it’s converted to:

@abbreviation\{\langle label \rangle, 
short = {\langle short \rangle}, 
long = {\langle long \rangle} \}


\newacronym

The base glossaries package provides:

\newacronym\[\langle key=value list \rangle\{\langle label \rangle\}{\langle short \rangle}{\langle long \rangle}\]

(which is redefined by glossaries-extra to use \newabbreviation).

As above but uses \acronym instead. The base package also provides \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 \acronym.
\texttt{\textbackslash glsxtrnewsymbol}\smallskip

The glossaries-extra package provides:

\begin{verbatim}
\texttt{\textbackslash glsxtrnewsymbol[\langle\text{key}=\text{value list}\rangle\langle\text{label}\rangle\langle\text{symbol}\rangle}
\end{verbatim}

(when the \texttt{symbols} option is used). This is converted to:

\begin{verbatim}
\@symbol\langle\text{label}\rangle,
    \texttt{name} = \langle\text{symbol}\rangle}
\end{verbatim}

if the optional argument is missing, otherwise it’s converted to:

\begin{verbatim}
\@symbol\langle\text{label}\rangle,
    \texttt{name} = \langle\text{symbol}\rangle,
    \langle\text{key}=\text{value list}\rangle
\end{verbatim}

unless \langle\text{key}=\text{value list}\rangle contains the \texttt{name} field, in which case it’s converted to:

\begin{verbatim}
\@symbol\langle\text{label}\rangle,
    \langle\text{key}=\text{value list}\rangle
\end{verbatim}

\texttt{newsym} is recognised as a synonym for \texttt{\textbackslash glsxtrnewsymbol}.

\texttt{\textbackslash glsxtrnewnumber}\smallskip

The glossaries-extra package provides:

\begin{verbatim}
\texttt{\textbackslash glsxtrnewnumber[\langle\text{key}=\text{value list}\rangle}\langle\text{label}\rangle}
\end{verbatim}

(when the \texttt{numbers} option is used). This is converted to:

\begin{verbatim}
\@number\langle\text{label}\rangle,
    \texttt{name} = \langle\text{label}\rangle}
\end{verbatim}

if the optional argument is missing, otherwise it’s converted to:

\begin{verbatim}
\@number\langle\text{label}\rangle,
    \texttt{name} = \langle\text{label}\rangle,
    \langle\text{key}=\text{value list}\rangle
\end{verbatim}

if \texttt{name} isn’t listed in \langle\text{key}=\text{value list}\rangle, otherwise it’s converted to:

\begin{verbatim}
\@number\langle\text{label}\rangle,
    \langle\text{key}=\text{value list}\rangle
\end{verbatim}

\texttt{newnum} is recognised as a synonym for \texttt{\textbackslash glsxtrnewnumber}.
7.2 Recognised Commands

\newdualentry

\newdualentry[⟨key=value list⟩]{⟨label⟩}{⟨short⟩}{⟨long⟩}{⟨description⟩}

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 
  {SVM}% abbreviation 
  {support vector machine}% long form 
  {Statistical pattern recognition technique}% description

then the .bib file will contain:


@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 \LaTeX{} 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 \LaTeX{} 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.

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 \sortop{} command described on 208 and need to be hidden from bib2gls’s interpreter. This file doesn’t contain any entry definitions and must be loaded first with 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:

\verbatim{
\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 (surname), (first name(s)) but for the glossary to display (first name(s)) (surname). For names with a “von” part, there’s another command:

\verbatim{
\sortvonname{(first name(s))}{(von)}{(surname)}
}

which has a similar purpose. The third command is:

\verbatim{
\sortart{(article)}{(text)}
}
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{⟨\text{first name(s)}⟩}{⟨\text{surname}⟩}

which again is functionally the same as `\sortname`.

The names could be specified using \LaTeX's syntax instead with `\bibtextcontributor-fields` to convert it, but the aim here is to show a variety of ways to use `bib2gls`. For an example of `\bibtextcontributor-fields`, see the way the `cast` field in `films.bib` is dealt with.

Although the file only contains ASCII characters, it starts with an encoding line to prevent `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 `no-interpret-preamble.bib` are as follows:

```latex
% 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}"
```

### interpret-preamble.bib

This provides definitions of `\sortname`, `\sortvonname`, `\sortart` and `\sortmediacreator` in `@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:

```latex
% Encoding: UTF-8
@preamble{"\providecommand{\sortname}[2]{#2, #1} \providecommand{\sortvonname}[3]{#2 #3, #1} \providecommand{\sortart}[2]{#2} \providecommand{\sortmediacreator}[2]{#2, #1}"
```

### interpret-preamble2.bib

An alternative to `interpret-preamble.bib` with a different definition of `\sortmediacreator`. This uses `\renewcommand` instead of `\providecommand` so `write-preamble={false}` is required to prevent \LaTeX{} from picking up the definitions.
The contents of interpret-preamble2.bib are as follows:

```latex
\% 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:

```plaintext
entry-type-aliases={constant=entry},
field-aliases=
  {constantname=name, 
   constantsymbol=symbol, 
   definition=description, 
   identifier=category, 
   value=user1 }
```

This setting means that, for example,

```plaintext
@constant{root2, 
  constantname={Pythagoras' constant}, 
  constantsymbol={\ensuremath{\sqrt{2}}}, 
  definition={the square root of 2}, 
  value={1.41421}, 
  identifier={constant} 
}
```

is treated as though it was defined as:

```plaintext
@entry{root2, 
  name={Pythagoras' constant}, 
  symbol={\ensuremath{\sqrt{2}}}, 
  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-aliases={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{\constanti}{\mathrm{i}}
\providecommand{\constantj}{\mathrm{j}}
\providecommand{\constante}{\mathrm{e}}
\providecommand{\constantpi}{\uppi}
\providecommand{\constantgamma}{\upgamma}
8 Examples: constants.bib

\providecommand{\constantphi}{\upphi}
\providecommand{\constantlambda}{\uplambda}
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
.glistex 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:

% 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},
@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
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}
}
```
@chemical{SO2, 
  formula={\ce{SO2}},
  chemicalname={sulfur dioxide},
  identifier={chemical}
}

@chemical{S2O72-, 
  formula={\ce{S2O7^{2-}}},
  chemicalname={disulfate ion},
  identifier={chemical}
}

@chemical{SbBr3, 
  formula={\ce{SbBr3}},
  chemicalname={antimony(III) bromide},
  identifier={chemical}
}

@chemical{Sc2O3, 
  formula={\ce{Sc2O3}},
  chemicalname={scandium oxide},
  identifier={chemical}
}

@chemical{Zr3P044, 
  formula={\ce{Zr3(PO4)4}},
  chemicalname={zirconium phosphate},
  identifier={chemical}
}

@chemical{ZnF2, 
  formula={\ce{ZnF2}},
  chemicalname={zinc fluoride},
  identifier={chemical}
}

bacteria.bib

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:
% Encoding: UTF-8

@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={Babesia burnedii},
  identifier={bacteria}
}
8 Examples: baseunits.bib

```latex
long={Coxiella burnetii},
identifier={bacteria}
}

@abbreviation{raustralis,
    short={R.-australis},
    long={Rickettsia australis},
    identifier={bacteria}
}

@abbreviation{rrickettsii,
    short={R.-rickettsii},
    long={Rickettsia rickettsii},
    identifier={bacteria}
}
```

**baseunits.bib**

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:

```latex
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:

```latex
% 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 \texttt{measurement}, which must be aliased otherwise the entries will all be ignored. The entries all have a custom field \texttt{identifier} set to \texttt{derivedunit}. This will be ignored by \texttt{bib2gls} unless it’s defined using \texttt{\glsaddkey} or \texttt{\glsaddstorage-key} or if it’s aliased with \texttt{field-aliases}.

The contents of derivedunits.bib are as follows:

\begin{verbatim}
% Encoding: UTF-8
% requires siunitx.sty

@measurement{area,
    unitname={square metre},
    unitsymbol={\si{\metre\text{\textsuperscript{2}}}},
    measurement={area},
    identifier={derivedunit}
}

@measurement{volume,
    unitname={cubic metre},
    unitsymbol={\si{\metre\text{\textsuperscript{3}}}},
    measurement={volume},
    identifier={derivedunit}
}

@measurement{velocity,
    unitname={metre per second},
    unitsymbol={\si{\metre\text{\textper{second}}}},
    measurement={velocity},
    identifier={derivedunit}
}

@measurement{acceleration,
    unitname={metre per second squared},
    unitsymbol={\si{\metre\text{\textper{square\text{second}}}}},
    measurement={acceleration},
    identifier={derivedunit}
}

@measurement{density,
    unitname={ampere per square metre},
    unitsymbol={\si{\ampere\text{\textper{square\text{metre}}}}},
    measurement={density},
    identifier={derivedunit}
}
\end{verbatim}
8 Examples: people.bib

@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 \texttt{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 \texttt{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 \texttt{born} field could be specified as:

\begin{verbatim}
born=\{\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 \texttt{\DTMdisplaydate} is provided by the \texttt{datetime2} package) and a definition could be provided for \texttt{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, \texttt{date-sort-format=\{d/M/y G\}}.

Some of the entries, such as \texttt{caesar}, have a \texttt{first} field. In those cases the \texttt{first} field is slightly different from the \texttt{name} field (for example, “Gaius” is omitted in \texttt{caesar}’s \texttt{first} field). The other entries don’t have a \texttt{first} field. They can simply have the \texttt{name} copied to \texttt{first} with the \texttt{replicate-fields} option (so that the full name is shown on first use) or the \texttt{first} field can be ignored with \texttt{ignore-fields} (so all entries will use the \texttt{text} field on first use). The \texttt{replicate-override} option can be used to force the \texttt{name} field to be copied to the \texttt{first} field, even if the \texttt{first} field is already set. Alternatively, with \texttt{replicate-override=\{true\}} and \texttt{replicate-fields=\{first=name\}}, the \texttt{first} field be copied to the \texttt{name} field. For consistency, the \texttt{first} fields use the same custom commands as used in the \texttt{name} field.

There’s one name with a “von” part. In this case the \texttt{name} field is set to:

\begin{verbatim}
\sortvonname{Manfred}{von}{Richthofen}
\end{verbatim}

which will come under the “V” letter group since \texttt{\sortvonname} is defined as \texttt{\{von\} \{surname\}, \{first name\(s)\}}.

If you prefer that this name should come under “R” instead, then you need to adjust the definition of \texttt{\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 \LaTeX{}’s contributor syntax and use \texttt{bibtex-contribution-fields=\{name\}} to convert them.

There are also some synonyms provided with \texttt{@index} entry types that have the \texttt{alias} field to redirect to the main entry. These don’t include a \texttt{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 \texttt{identifier} set to \texttt{person}. This will be ignored by \texttt{bib2gls} unless it’s defined using \texttt{\glsaddkey} or \texttt{\glsaddstoragekey} or if it’s aliased with \texttt{field-aliases}.

The contents of \texttt{people.bib} are as follows:
8 Examples: people.bib

% Encoding: UTF-8

@entry{caesar,
    name={\sortname{Gaius Julius}{Caesar}},
    first={\sortname{Julius}{Caesar}},
    text={Caesar},
    description={Roman politician and general},
    born={13-July 100 BC},
    died={15-March 44 BC},
    identifier={person}
}

@entry{wellesley,
    name={\sortname{Arthur}{Wellesley}},
    text={Wellington},
    description={Anglo-Irish soldier and statesman},
    born={1-May 1769 AD},
    died={14-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-July 1769 AD},
    died={5-May 1821 AD},
    identifier={person}
}

@entry{alexander,
    name={Alexander III of Macedon},
    text={Alexander},
    description={Ancient Greek king of Macedon},
    born={20-July 356 BC},
    died={10-June 323 BC},
    othername={Alexander the Great},
    identifier={person}
}

338
@index{alexanderthegreat,
  name={Alexander the Great},
  alias={alexander},
  identifier={person}
}

@entry{vonrichthofen,
  name={\sortvonname{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},}
@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},
    identifier={person}
examples: books.bib

books.bib

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 \texttt{ Bib\TeX}’s syntax instead with \texttt{bibtex-contributor-fields} to convert it. Alternatively, the entries could be defined using standard \texttt{ Bib\TeX} entry types that are all aliased to \texttt{@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}},
    
    \textbf{342}
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}
}

343
@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-interp-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
8 Examples: films.bib

(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 principle 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:

\texttt{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:

\texttt{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:

\texttt{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, \texttt{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 \texttt{glsxtrrestorepostpunc} for use in the category post-description hooks that counteracts \texttt{glsxtrnopostpunc}. This can be placed inside a conditional, as used in sample-media.tex, and does nothing if \texttt{glsxtrnopostpunc} 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}:  
\sortart{The}{Two Towers}}
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},
}
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 BibTeX file, but it can be parsed by bib2gls if the BibTeX entry types are converted to @bibtexentry, which can easily be done with:

entry-type-aliases={\GlsXtrBibTeXEntryAliases}

The field names will also need to be defined or aliased. For example:

field-aliases={title=name}

If bib2gls is then run with --cite-as-record any \citation commands found in the .aux file will be treated as ignored records. The @preamble provides a formatting command that’s used by both BibTeX and bib2gls, so \providecommand is required rather than \newcommand as it will appear in both the .bb1 and the .glstex files. (In general it’s best to use \providecommand rather than \newcommand in the @preamble but in this case it’s essential.)

The contents of citations.bib are as follows:

% Encoding: UTF-8

@preamble{"\providecommand{\titlefmt}[1]{`#1'}"}

@article{duck2018,
  author = {Dickie Duck and José 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
}
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 TeX parser library recognises all the mathematical Greek letter commands provided in the E\TeX{} kernel. Additionally it recognises \texttt{\omicron} which isn’t provided by E\TeX{} (the symbol can be reproduced with a lower case Latin “o”). Note that glossaries-extra-bib2gls (glossaries-extra v1.27+) provides all the missing Greek letters (such as \texttt{omicron}).

The .bib file could just use o:

\texttt{@symbol\{omicron,}
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 L\LaTeX{} needs a definition for this, so it’s provided in the \texttt{@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 \texttt{mathgreek.bib} are as follows:

```latex
% 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={\(\varepsilon\) (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}
}
8 Examples: mathgreek.bib

@symbol{\mu, 
  name={\texttt{\textbackslash ensuremath\{\mu\}}}, 
  description={\mu}, 
  identifier={mathgreek}}

@symbol{\nu, 
  name={\texttt{\textbackslash ensuremath\{\nu\}}}, 
  description={\nu}, 
  identifier={mathgreek}}

@symbol{\xi, 
  name={\texttt{\textbackslash ensuremath\{\xi\}}}, 
  description={\xi}, 
  identifier={mathgreek}}

@symbol{\omicron, 
  name={\texttt{\textbackslash ensuremath\{\omicron\}}}, 
  description={\omicron}, 
  identifier={mathgreek}}

@symbol{\pi, 
  name={\texttt{\textbackslash ensuremath\{\pi\}}}, 
  description={\pi}, 
  identifier={mathgreek}}

@symbol{\rho, 
  name={\texttt{\textbackslash ensuremath\{\rho\}}}, 
  description={\rho}, 
  identifier={mathgreek}}

@symbol{\varsigma, 
  name={\texttt{\textbackslash ensuremath\{\varsigma\}}}, 
  description={\sigma (variant)}, 
  identifier={mathgreek}}

@symbol{\sigma, 
  name={\texttt{\textbackslash ensuremath\{\sigma\}}}, 
  description={\sigma}, 
  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]{$#1$-ary}"} 

The first command $\bigoperatornamefmt{(text)}$ 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
8 Examples: bigmathsymbols.bib

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 \nary\{<text>\} to provide semantic markup for “n-ary”. This could be defined without an argument:

\providecommand{\nary}{$n$-ary}  

but providing an argument will allow \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:

% Encoding: UTF-8

% requires stix.sty

@preamble{"\providecommand{\bigoperatornamefmt}[1]{}\"}
\providecommand{\nary}[1]{$#1$-ary"

@symbol{bigsqcap,
  name={\bigoperatornamefmt{bigsqcap}},
  text={bigsqcap},
  description={\nary{n} square intersection operator},
  identifier={naryoperator} }

@symbol{bigsqcup,
  name={\bigoperatornamefmt{bigsqcup}},
  text={bigsqcup},
  description={\nary{n} square union operator},
  identifier={naryoperator} }

@symbol{sum,
  name={\bigoperatornamefmt{sum}},
  text={sum},
  description={\nary{n} summation},
  identifier={naryoperator} 358
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={\nary{n} circled times operator},
identifier={naryoperator}
}

@symbol{biguplus,
   name={\bigoperatornamefmt{\biguplus}},
   text={\biguplus},
description={\nary{n} union operator with plus},
identifier={naryoperator}
}

@symbol{bigvee,
   name={\bigoperatornamefmt{\bigvee}},
   text={\bigvee},
description={\nary{n} logical or},
identifier={naryoperator}
}

@symbol{bigwedge,
   name={\bigoperatornamefmt{\bigwedge}},
   text={\bigwedge},
description={\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}
}
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{leq}` 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:

```plaintext
% 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}}}
```
@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={$\approx$},  
  description={approx},  
  identifier={relation}  
}
binaryoperators.bib

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:

% Encoding: UTF-8

@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}}
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:

% 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 \vert not | as in some of the earlier examples.)
\numspacefmt{⟨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 variable(s)\rangle}{\langle condition\rangle}}

The simplest thing to do here is to provide a wrapper command in the document, for example:

\newcommand*{\setmembership}[2]{\glsxtrfmt{setmembership}{{#1}{#2}}}

Now this can be used as:

\setmembership{\langle variable(s)\rangle}{\langle 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 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 identifier field that reflects the type of object: set for sets, numberspace for number spaces, 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 \gls or \glsxtrfmt within link text and that includes within the \text argument of \glsxtrfmt. See sample-maths.tex for suggested usage.

Some of the description fields use \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}  
}
@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={the cardinality of $\setfmt{S}$},
  format={setcardfmt},
  identifier={set}
}

@symbol{numberspace, 
  name={\ensuremath{\numspacefmt{S}}},
  description={a number space},
  format={numspacefmt},
  identifier={numberspace}
}

@symbol{naturalnumbers, 
  name={\ensuremath{\numspacefmt{N}}},
  description={the set of natural numbers},
  identifier={numberspace}
}

@symbol{integernumbers, 
  name={\ensuremath{\numspacefmt{Z}}},
  description={the set of integers},
  identifier={numberspace}
}

@symbol{rationalnumbers, 
  name={\ensuremath{\numspacefmt{Q}}},
  description={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 $\matxfmt{M}$}},
format={invfmt},
identifier={matrix}
}

@symbol{vector,
   name={\ensuremath{\vecfmt{v}}},
   description={\Sortart{a}{vector}},
   format={vecfmt},
   identifier={matrix}
}

@symbol{matrix,
   name={\ensuremath{\matxfmt{M}}},
   description={\Sortart{a}{matrix}},
   format={matxfmt},
   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{\matxfmt{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:
This removes the conflict, and `\Sun` and `\Lightning` are as defined by \texttt{marvosym}.

This file uses a custom entry type \texttt{@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 \texttt{identifier} field set to \texttt{mediacontrol}, \texttt{information} and \texttt{weather}, respectively. There are two other custom fields: \texttt{icon} and \texttt{icondescription}. These will need to be aliased to \texttt{name} and \texttt{description}.

Neither of these packages are recognised by \texttt{bib2gls}, which means that \texttt{set-widest} won’t be able to determine the widest name nor is this data suitable for sorting according to the \texttt{icon} field (or its alias). Instead, either sort by label (which is the default for \texttt{@symbol}) or by the \texttt{description}. If you want to use one of the alttree styles you can still use \texttt{set-widest}, but it will have to use the fallback command. Alternatively, you can omit \texttt{set-widest} and explicitly use \texttt{\glsFindWidestTopLevelName}.

The contents of \texttt{miscsymbols.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: micsymbols.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={cafe},
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}
}
8 Examples: miscsymbols.bib

@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},  
}
8 Examples: markuplanguages.bib

```bibtex
identifier={weather}
}

@icon{suncld, icon={\SunCloud}, icondescription={overcast}, identifier={weather}
}

@icon{raincloud, icon={\RainCloud}, icondescription={rain}, identifier={weather}
}

@icon{weakraincloud, icon={\WeakRainCloud}, icondescription={drizzle}, identifier={weather}
}

@icon{snowcloud, icon={\SnowCloud}, icondescription={snow}, identifier={weather}
}
```

**markuplanguages.bib**

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:

```bibtex
% 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 \glstext{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={xtensible \markuplang},
  description={a markup language that defines a set of rules for
            encoding documents},
  identifier={markuplanguage}
}

@abbreviation{html,
  short={HTML},
  long={\hyphertext \markuplang},
  description={the standard markup language for creating web pages},
  identifier={markuplanguage}
}

@abbreviation{mathml,
  short={MathML},
  long={\NoCaseChange{m}athematical \markuplang},
  description={markup language for describing mathematical notation},
  identifier={markuplanguage}
}
@abbreviation{xhtml,
    short={XHTML},
    long={extensible \ hyper\text }
    # markuplang,
    description={\xml version=html},
    identifier={markuplanguage}
}

@abbreviation{svg,
    short={SVG},
    long={scalable \ vector \ graphics},
    description={\xml-based vector image format},
    identifier={markuplanguage}
}

usergroups.bib

The usergroups.bib file requires either XƎL ATEX or LuaL ATEX as some of the entry labels use non-ASCII characters. This file has a mixture of @abbreviation and @index entries. It also uses @string for convenience and provides a custom command \dash in @preamble. Each entry is the name of a \TeX user group: the international \TeX Users Group (TUG) and all the local groups. Most of them have an abbreviated name, so they’re defined with @abbreviation. There are a few without an abbreviation, so they’re defined with @index instead. There’s one alias. (The information was obtained from TUG’s user groups page [18].)

As with the other examples, there are some custom fields which will be ignored if they aren’t defined or aliased: identifier (set to \textusergroup), language (a comma-separated list of language tags) and translation (provides a translation if the user group name isn’t in English).

Not all entries have a translation field. It it’s omitted, then the user group name is in English, otherwise it’s in the first language listed in the 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:

% Encoding: UTF-8

% Requires XeLaTeX/LuaLaTeX for non-ASCII labels

@string{tug={\TeX \ Users Group}}

@preamble{"\providecommand{\dash}{\,---\,}"}

@abbreviation{TUG,
    short={TUG},
}
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}
8 Examples: usergroups.bib

@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,


8 Examples: animals.bib

language={vi},
identifier={texusergroup}
}

@abbreviation{LUGSA,
    short={LUGSA},
    long={\LaTeX\ User Group\dash South Africa},
    language={en-ZA},
    identifier={texusergroup}
}

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:

% Encoding: UTF-8

@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}},
    identifier={animal}
}
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\gls{quartz}},
   identifier={mineral}
}

@entry{chalcedony,
   name={chalcedony},
   description={cryptocrystalline variety of\gls{quartz}},
   identifier={mineral}
}

@entry{citrine,
   name={citrine},
   description={yellow variety of\gls{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

383
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:
% Encoding: UTF-8

@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},
8 Examples: terms.bib

```latex
\begin{verbatim}
alias={courgette},
identifier={vegetable}
\end{verbatim}

terms.bib

The terms.bib file contains entries defined using \texttt{@index}. Unlike the above sample .bib files, there are no custom fields here.

The contents of terms.bib are as follows:

\begin{verbatim}
% 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}
\end{verbatim}
```
8 Examples: topics.bib

@index{measurement}

topics.bib

The topics.bib file contains entries defined using @index. Again there are no custom fields here.

The contents of topics.bib are as follows:

% Encoding: UTF-8

@index{information}
@indexplural{mediacontrol,text={media control}}
@indexplural{weather,text={weather symbol}}

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:

src={terms,animals,minerals,vegetables}

The custom identifier field is aliased to the parent field since it conveniently matches the labels of the animal, mineral and vegetable entries in the terms.bib file:

field-aliases={identifier=parent}

The default 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 \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 \bib2gls+\LaTeX they will also have location lists. This would make the complete document build:

\pdf\LaTeX sample-hierarchical
\bib2gls --group sample-hierarchical
\pdf\LaTeX sample-hierarchical
\bib2gls --group sample-hierarchical
\pdf\LaTeX sample-hierarchical

However, in this example I’ve decided to ignore any records created in the glossary:
8 Examples: sample-nested.tex

\GlsXtrSetDefaultNumberFormat{glsignore}

This means that the document build is the usual \LaTeX+\bib2gls+\LaTeX.

I’ve used the treeseq 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:

pdf\LaTeX sample-hierarchical
\bib2gls --group sample-hierarchical
pdf\LaTeX sample-hierarchical

The complete document is shown in figure 8.1.

\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=treeseq]{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}

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 type, group and parent fields. In general, hierarchical
glossaries are created with the parent field and an appropriate glossary style (as in the
previous 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.
  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:

```latex
@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:

```latex
@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 \ifglossary-exists to included ignored glossaries in the existence check. If the glossary exists, it’s then displayed using \printunsrtinnerglossary:

```latex
\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 \texttt{\printunsrtglossary} handler macro then needs to be set to this custom macro when the main glossary is displayed:

\begin{verbatim}
\printunsrtglossary*{\let\printunsrtglossaryhandler\nestedhandler}
\end{verbatim}

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:

\begin{verbatim}
\GlsXtrLoadResources[src={terms,animals,minerals,vegetables},
    field-aliases={identifier=type},
    dependency-fields={type}]
\end{verbatim}

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:

\begin{verbatim}
\GlsXtrLoadResources[src={terms},sort={en-reverse}]
\GlsXtrLoadResources[src={animals,minerals,vegetables},sort={en},
    field-aliases={identifier=type},dependency-fields={type}]
\end{verbatim}

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:
Fortunately in this example, cross-resource dependencies are permitted so the simpler alternative works. (If they’re not permitted, the bib2gls transcript file will contain “Cross-resource references can’t be supported for resource set ⟨filename⟩”.)

The complete code is listed below. The document build is:

```latex
\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}]

newcommand{\nestedhandler}[1]{
  \glsxtrunsrtdo{#1}% Is there a glossary whose label (type) matches this entry’s label?
}
```

The complete document is shown in figure 8.2.
\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}

\begin{document}

\section*{sample-constants.tex}

This example uses the \texttt{constants.bib} file. The aim here is to just have a list of all the constants defined in the \texttt{.bib} file. (There are no references in the document.) This means I need to use:

\begin{verbatim}
selection={all}
\end{verbatim}

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:

\begin{verbatim}
entry-type-aliases={constant=number}
\end{verbatim}

The custom fields also need aliasing:

\begin{verbatim}
field-aliases=
    {identifier=category, constantsymbol=name, constantname=description, value=user1, definition=user2, alternative=user3,
    }
\end{verbatim}

\end{document}
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
marrow long white-fleshed gourd with green skin.

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.
quartzite 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

1

Figure 8.2: sample-nested.pdf
I decided to use the altlist style, so I’ve instructed bib2gls to determine the widest name:

\setwidest

It’s always a good idea to specify the glossary type when using \setwidest, although in this example there’s only one glossary so it doesn’t make much difference.

\type{main}

I decided to order the constants according to their (approximate) numerical value. I’ve aliased the custom value field to \texttt{user1}, so I can sort by that field using a numerical comparison:

sort-field={user1},
\sort{double}

There are three entries without the \texttt{user1} field (as the custom 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 name field. Since I’ve change the default sort-field, I can’t use symbol-sort-fallback. Instead I need to use:

\missing-sort-fallback={name}

What happens with the imaginary entry? It has no real representation. The transcript (.glg) file shows the message:

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 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 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:

\usepackage[record,nostyles,postdot,stylemods=tree,style=alttree]
\{glossaries-extra\}

I’ve assigned the custom constantname field to the description field and the custom constantsymbol field to the 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(category), so I need to define the command \glsxtrpostdescconstant. A simple definition is:

```latex
\newcommand{\glsxtrpostdescconstant}{%
  \space (approximately \text{\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:

```latex
\newcommand{\glsxtrpostdescconstant}{%
  \ifglshasfield{user1}{\glscurrententrylabel}%
  { (approximately \text{\glscurrentfieldvalue})}%
  {}%
  \ifglshasfield{user2}{\glscurrententrylabel}%
  { : \text{\glscurrentfieldvalue}}%
  {}%
%
}%
```

or with \glsxtrifhasfield:

```latex
\newcommand{\glsxtrpostdescconstant}{%
  \glsxtrifhasfield{useri}{\glscurrententrylabel}%
  { (approximately \text{\glscurrentfieldvalue})}%
  {}%
  \glsxtrifhasfield{userii}{\glscurrententrylabel}%
  { : \text{\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):

```latex
\newcommand{\glsxtrpostdescconstant}{%
  \glsxtrifhasfield{useriii}{\glscurrententrylabel}%
  { (also denoted \text{\glscurrentfieldvalue}}%
  \glsxtrifhasfield{useri}{\glscurrententrylabel}%
  {, approximately \text{\glscurrentfieldvalue}}%
  {}%
  \glsxtrifhasfield{userii}{\glscurrententrylabel}%
  {}%
  \glsxtrifhasfield{useriii}{\glscurrententrylabel}%
  {, approximately \text{\glscurrentfieldvalue}}%
  {}%
  \glsxtrifhasfield{useri}{\glscurrententrylabel}%
  { : \text{\glscurrentfieldvalue}}%
  {}%
  \glsxtrifhasfield{userii}{\glscurrententrylabel}%
  { : \text{\glscurrentfieldvalue}}%
  {}%
  \glsxtrifhasfield{useriii}{\glscurrententrylabel}%
  { : \text{\glscurrentfieldvalue}}%
  {}%
%
}%
```
8 Examples: sample-constants.tex

{%
\glsxtrifhasfield{useri}{\glscurrententrylabel}{%
{ (approximately \glscurrentfieldvalue)}%
{}%
}\glsxtrifhasfield{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 \texttt{glsxtrpostdesc} part of the command name.

The complete code is listed below. The document build is:

pdflatex sample-constants
bib2gls sample-constants
pdflatex 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,
\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:

save-locations={false}

All entries are defined in chemicalformula.bib using a custom entry type @chemical which needs to be aliased in order for the entries to be recognised:

entry-type-aliases={chemical=symbol}
8 Examples: sample-chemical.tex

Constants

\begin{itemize}
  \item \(i\): imaginary unit (also denoted \(j\)); defined as \(i^2 = -1\).
  \item \(0\): nothing or nil.
  \item \(\gamma\): Euler’s constant (approximately 0.57721): the limit of
    \[\sum_{r=1}^{n} \frac{1}{r} - \ln n\]
    as \(n \to \infty\).
  \item \(1\): single entity, unity.
  \item \(\zeta(3)\): Apéry’s constant (approximately 1.2020569): a special value of the Riemann zeta function.
  \item \(\lambda\): Conway’s constant (approximately 1.30357): the invariant growth rate of all derived strings.
  \item \(\sqrt{2}\): Pythagoras’ constant (approximately 1.41421): the square root of 2.
  \item \(\phi\): golden ratio (approximately 1.61803): the ratio \(\frac{1+\sqrt{5}}{2}\).
  \item \(e\): Euler’s number (approximately 2.71828): base of natural logarithms.
  \item \(\pi\): pi (approximately 3.14159): the ratio of the length of the circumference of a circle to its diameter.
\end{itemize}
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:

```latex
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:

```latex
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:

```latex
\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:

```latex
\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:

```latex
pdflatex sample-chemical
bib2gls --group sample-chemical
pdflatex sample-chemical
```

The complete document is shown in figure 8.4.

```latex
\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}
```
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*{\glstreenamefmt}[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]{\textit{#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: Al\textsubscript{2}(SO\textsubscript{4})\textsubscript{3}, H\textsubscript{2}O, C\textsubscript{6}H\textsubscript{12}O\textsubscript{6}, CH\textsubscript{3}CH\textsubscript{2}OH, CH\textsubscript{2}O, OF\textsubscript{2}, O\textsubscript{2}F\textsubscript{2}, SO\textsubscript{4}\textsuperscript{2−}, H\textsubscript{3}O\textsuperscript{+}, OH\textsuperscript{−}, O\textsubscript{2}, AlF\textsubscript{3}, O, Al\textsubscript{2}CsO\textsubscript{3}, As\textsubscript{4}S\textsubscript{4}, C\textsubscript{10}H\textsubscript{10}O\textsubscript{4}, C\textsubscript{5}H\textsubscript{4}NCOOH, C\textsubscript{4}H\textsubscript{12}N\textsubscript{4}O\textsubscript{2}, SO\textsubscript{2}, S\textsubscript{2}O\textsubscript{7}\textsuperscript{2−}, SbBr\textsubscript{3}, Sc\textsubscript{2}O\textsubscript{3}, Zr\textsubscript{3}(PO\textsubscript{4})\textsubscript{4}, ZnF\textsubscript{2}.

Glossary

<table>
<thead>
<tr>
<th>A</th>
<th>aluminium trifluoride</th>
</tr>
</thead>
<tbody>
<tr>
<td>Al\textsubscript{2}(SO\textsubscript{4})\textsubscript{3}</td>
<td>aluminium sulfate</td>
</tr>
<tr>
<td>Al\textsubscript{2}CoO\textsubscript{4}</td>
<td>cobalt blue</td>
</tr>
<tr>
<td>As\textsubscript{4}S\textsubscript{4}</td>
<td>tetraarsenic tetrasulfide</td>
</tr>
<tr>
<td>C</td>
<td>formaldehyde</td>
</tr>
<tr>
<td>CH\textsubscript{3}CH\textsubscript{2}OH</td>
<td>ethanol</td>
</tr>
<tr>
<td>C\textsubscript{6}H\textsubscript{12}NCOOH</td>
<td>niacin</td>
</tr>
<tr>
<td>C\textsubscript{5}H\textsubscript{4}NCOOH</td>
<td>niacin</td>
</tr>
<tr>
<td>C\textsubscript{8}H\textsubscript{10}N\textsubscript{4}O\textsubscript{2}</td>
<td>caffeine</td>
</tr>
<tr>
<td>C\textsubscript{10}H\textsubscript{10}O\textsubscript{4}</td>
<td>ferulic acid</td>
</tr>
<tr>
<td>H</td>
<td>water</td>
</tr>
<tr>
<td>H\textsubscript{3}O\textsuperscript{+}</td>
<td>hydronium</td>
</tr>
<tr>
<td>O</td>
<td>oxygen</td>
</tr>
<tr>
<td>OF\textsubscript{2}</td>
<td>oxygen difluoride</td>
</tr>
<tr>
<td>OH\textsuperscript{−}</td>
<td>hydroxide ion</td>
</tr>
<tr>
<td>O\textsubscript{2}</td>
<td>dioxygen</td>
</tr>
<tr>
<td>O\textsubscript{2}F\textsubscript{2}</td>
<td>dioxygen difluoride</td>
</tr>
<tr>
<td>S</td>
<td>sulfur dioxide</td>
</tr>
<tr>
<td>SO\textsubscript{4}\textsuperscript{2−}</td>
<td>sulfate</td>
</tr>
<tr>
<td>S\textsubscript{2}O\textsubscript{7}\textsuperscript{2−}</td>
<td>disulfate ion</td>
</tr>
<tr>
<td>SbBr\textsubscript{3}</td>
<td>antimony(III) bromide</td>
</tr>
<tr>
<td>Sc\textsubscript{2}O\textsubscript{3}</td>
<td>scandium oxide</td>
</tr>
<tr>
<td>Z</td>
<td>zinc fluoride</td>
</tr>
<tr>
<td>Zr\textsubscript{3}(PO\textsubscript{4})\textsubscript{4}</td>
<td>zirconium phosphate</td>
</tr>
</tbody>
</table>

Figure 8.4: sample-chemical.pdf
attributes. The font used for the name in the glossary is governed by the glossnamefont attribute, the font used for the description in the glossary is governed by the glossdescfont attribute and the font used by commands like \gls in the document is governed by the text-format 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 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.¹ This isn’t loaded automatically, but it can be loaded through the stylemods package option:

\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 description field (which means I don’t need the glossdescfont attribute) and, since the long-only-short-only style sets the 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 \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:

¹glossary-bookindex is distributed with glossaries-extra v1.21+.
\begin{quote}
8 Examples: sample-bacteria.tex
\end{quote}

\renewcommand*\{\glsxtrbookindexname\}[1]{%
\glscategory{\#1}{bacteria}
{\glsentrynameother{\#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 \langle long \rangle (\langle short \rangle) 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}[1]{\texttt{#1}}
```
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{\textbackslash gls\textbackslash xtrsetgrouptitle\{baseunits\}\{Base Units\}}
\texttt{\textbackslash gls\textbackslash xtrsetgrouptitle\{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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B</strong></td>
<td><strong>C</strong></td>
</tr>
<tr>
<td>Bacillus subtilis, 1</td>
<td>Clostridium botulinum, 1</td>
</tr>
<tr>
<td></td>
<td>Clostridium perfringens, 1</td>
</tr>
<tr>
<td></td>
<td>Clostridium tetani, 1</td>
</tr>
<tr>
<td></td>
<td>Coxiella burnetii, 1</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>R</strong></td>
</tr>
<tr>
<td>Planifilum composti, 1</td>
<td>Rickettsia australis, 1</td>
</tr>
<tr>
<td>Planifilum fimeticola, 1</td>
<td>Rickettsia rickettsii, 1</td>
</tr>
<tr>
<td>pseudo monas putida, 1</td>
<td></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 \texttt{.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.
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:

```
entry-type-aliases={unit=symbol}
```

for the first resource set and

```
entry-type-aliases={measurement=symbol}
```

for the second. Similarly for the custom entry fields:

```
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:

```
pdflatex sample-units2
bib2gls --group sample-units2
pdflatex sample-units2
```

The complete document is shown in figure 8.7.

```
\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},
```

410
8 Examples: sample-units3.tex

\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:

\begin{verbatim}
field-aliases={
    unitname=name,
    unitsymbol=symbol,
    measurement=description
},
symbol-sort-fallback=name,
selection={all}
}
\GlsXtrLoadResources[
    src={derivedunits},
    type=derivedunits,
    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}
\end{verbatim}

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 \texttt{bib2gls} to make the custom \texttt{@unit} and \texttt{@measurement} entry types behave like \texttt{@dualsymbol}.
8 Examples: sample-units3.tex

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 (m$^3$) volume

cubic metre per kilogram (m$^3$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 (m$^2$) area

Figure 8.7: sample-units2.pdf
8 Examples: sample-units3.tex

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:

\begin{verbatim}
\setlength{\glsdescwidth}{.4\hsize}
\end{verbatim}

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:

\begin{verbatim}
\newglossarystyle{units}{% style name
  \setglossarystyle{long3col-booktabs}%
  \renewcommand*{\glossaryheader}{%\toprule
    \bfseries Name & \bfseries Measurement & \bfseries Symbol
  \midrule
  \bottomrule\endhead
  \bottomrule\endfoot
  \% main entries:
  \renewcommand{\glossentry}[2]{%\glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
\end{verbatim}

413
There are no sub-entries in this document so I haven’t bothered to redefine \glossentry. (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.
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= inflict-numeric-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
    \textbf{Name} & \textbf{Measurement} & \textbf{Symbol} \\
    \midrule
    \bottomrule
  }%

  % main entries:
  \renewcommand{\glossentry}[2]{%
    \glossentryitem{##1}{\glstarget{##1}{\glossentryname{##1}}} &
    \glossentrydesc{##1}{\glspostdescription} &
    \glossentriesymbol{##1}{\tabularnewline
  }%
}
%
\begin{document}
\printunsrtglossary[title={SI Units of Measurement},
  style={units}]
\printunsrtglossary[type=units]
\end{document}
SI Units of Measurement

<table>
<thead>
<tr>
<th>Name</th>
<th>Measurement</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ampere</td>
<td>electric current</td>
<td>A</td>
</tr>
<tr>
<td>ampere per square metre</td>
<td>density</td>
<td>A m⁻²</td>
</tr>
<tr>
<td>candela</td>
<td>luminous intensity</td>
<td>cd</td>
</tr>
<tr>
<td>candela per square metre</td>
<td>luminance</td>
<td>cd m⁻²</td>
</tr>
<tr>
<td>cubic metre</td>
<td>volume</td>
<td>m³</td>
</tr>
<tr>
<td>cubic metre per kilogram</td>
<td>specific volume</td>
<td>m³ kg⁻¹</td>
</tr>
<tr>
<td>kelvin</td>
<td>thermodynamic temperature</td>
<td>K</td>
</tr>
<tr>
<td>kilogram</td>
<td>mass</td>
<td>kg</td>
</tr>
<tr>
<td>metre</td>
<td>length</td>
<td>m</td>
</tr>
<tr>
<td>metre per second</td>
<td>velocity</td>
<td>m s⁻¹</td>
</tr>
<tr>
<td>metre per second squared</td>
<td>acceleration</td>
<td>m s⁻²</td>
</tr>
<tr>
<td>mole</td>
<td>amount of substance</td>
<td>mol</td>
</tr>
<tr>
<td>mole per cubic metre</td>
<td>concentration</td>
<td>mol m⁻³</td>
</tr>
<tr>
<td>per metre</td>
<td>more standard</td>
<td>m⁻¹</td>
</tr>
<tr>
<td>second</td>
<td>time</td>
<td>s</td>
</tr>
<tr>
<td>square metre</td>
<td>area</td>
<td>m²</td>
</tr>
</tbody>
</table>

Units of Measurement (by SI unit)

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ampere</td>
</tr>
<tr>
<td>A m⁻²</td>
<td>ampere per square metre</td>
</tr>
<tr>
<td>cd</td>
<td>candela</td>
</tr>
<tr>
<td>cd m⁻²</td>
<td>candela per square metre</td>
</tr>
<tr>
<td>m³ kg⁻¹</td>
<td>cubic metre per kilogram</td>
</tr>
<tr>
<td>K</td>
<td>kelvin</td>
</tr>
<tr>
<td>kg</td>
<td>kilogram</td>
</tr>
<tr>
<td>m</td>
<td>metre</td>
</tr>
<tr>
<td>m s⁻¹</td>
<td>metre per second</td>
</tr>
<tr>
<td>m s⁻²</td>
<td>metre per second squared</td>
</tr>
<tr>
<td>mol</td>
<td>mole</td>
</tr>
<tr>
<td>mol m⁻³</td>
<td>mole per cubic metre</td>
</tr>
<tr>
<td>m⁻¹</td>
<td>per metre</td>
</tr>
<tr>
<td>s</td>
<td>second</td>
</tr>
</tbody>
</table>

Figure 8.8: sample-units3.pdf

sample-media.tex

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 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 (film.thebigsleep comes before thebigsleep) and sometimes having the book listed first (brightonrock comes before film.brightonrock).

One possible solution would be to also assign prefixes for the book labels, but 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 category field needs to be set. There are two simple ways to achieve this: use category={same as base} (which sets the category to books for entries in books.bib and to films for entries in films.bib) or alias the custom identifier field to category. I’ve chosen the latter method and also provided aliases for the custom year and cast fields:

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 `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 `category` I can provide semantic formatting commands (as for `sample-bacteria.tex`):

\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}

I’ve given films a slightly different format to make them easier to distinguish from books of the same name.

Both `books.bib` and `films.bib` had the custom `year` field, indicating the year of first publication or release, which I’ve assigned to the `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:

\newcommand*{\glsxtrpostnamebook}{\ifglshasfield{user1}{\glscurrententrylabel}{\space(published \glscurrentfieldvalue)}{}}
\newcommand*{\glsxtrpostnamefilm}{\ifglshasfield{user1}{\glscurrententrylabel}{\space(released \glscurrentfieldvalue)}{}}

As with the post-description hook, if you have at least `glossaries-extra v1.31`, it’s better to use:

\glsdefpostname{⟨category⟩}

instead of:

\newcommand{\glsxtrpostname⟨category⟩}

as it can guard against accidental misspelling of the `glsxtrpostname` part of the command name.

I’ve assigned the `cast` field to the `user2` field, and since this field uses \TeX’\textsuperscript{’}s contributor markup I need to convert this to a form that’s easier to customize:
bibtex-contributor-fields={user2}

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 cast 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⟨category⟩ to append the cast list:

\newcommand*{\glsxtrpostdescfilm}{\%\ifglshasfield{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*{\DTListformatOxford}{,}
\renewcommand*{\DTLandName}{and}

\newcommand*{\bookfont}[1]{\textbf{\emph{#1}}}
\newcommand*{\filmfont}[1]{\textsf{\textit{#1}}}
\glssetcategoryattribute{book}{textformat}{\bookfont}
\glssetcategoryattribute{book}{glossnamefont}{\bookfont}
\glssetcategoryattribute{film}{textformat}{\filmfont}
\glssetcategoryattribute{film}{glossnamefont}{\filmfont}

\newcommand*{\glsxtrpostnamebook}{%
  \ifglshasfield{user1}{\glscurrententrylabel}%
  \{space(published \glscurrentfieldvalue)}%
  
}

\newcommand*{\glsxtrpostnamefilm}{%
  \ifglshasfield{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}
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:

\texttt{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\textit{(category)}. 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:

\texttt{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:

\begin{verbatim}
\newcommand*{\glsxtrpostlinkpeople}{% 
  \ifglshasfield{user3}{\glslabel}%
    {\space(\glscurrentfieldvalue)}%
  {}%

}%

\newcommand*{\glsxtrpostnamepeople}{%
  \ifglshasfield{user3}{\glscurrententrylabel}%
    {\space(\glscurrentfieldvalue)}%
  {}%
}
\end{verbatim}

(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:
8 Examples: sample-people.tex

\newcommand*{\glstextrpostdescpeople}{%
  \ifglshasfield{user1}{\glscurrententrylabel}
  {\% born
    \space(\glscurrentfieldvalue\,--\,}%
  \ifglshasfield{user2}{\glscurrententrylabel}
  {\% died
    \glscurrentfieldvalue
  }%
  {}%
%}
%
%
%
%
}

The first list is quite straight-forward 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 sort option and there’s no document language, so bib2gls will sort according to my locale. The custom commands \sortname and \sortvonname ensure that the entries are all sorted alphabetically according to the surnames.

The second list can easily be created by adding the secondary option:

secondary={date:user1:bybirth}

This sorts according to the user1 field (which was originally the 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 JRE to using the CLDR you may find a change in the default format. In case the format provided in the .bib file isn’t recognised, the required format can be set with:

secondary-date-sort-format={d MMM YYYY G}

I’ve changed the date group headings by redefining \bibglsdategroup and \bibglsdategrouptitle, which means that the grouping in the bybirth glossary will be in the form ⟨year⟩ ⟨era⟩:

\newcommand{\bibglsdategroup}[7]{#1#4#7}
\newcommand{\bibglsdategrouptitle}[7]{\number#1 \ #4}

I’ve also defined the bybirth glossary and supplied a title:

423
The first two glossaries have entries with fairly long names (especially those with the post-name hook), 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:

\newglossary*{bybirth}{People (Ordered by Birth)}

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:
stylemods={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:

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 `.glstex` file by `bib2gls`, so I can’t use `\renewcommand` (although I could use `\glsrenewcommand`). Instead I’ve provided some custom commands:

\begin{verbatim}
\newcommand*{\swaptwo}[2]{#2, #1}
\newcommand*{\swapthree}[3]{#2 #3, #1}
\end{verbatim}

Now I just need to make an assignment using `\let`:

\begin{verbatim}
\let\sortname\swaptwo
\let\sortart\swaptwo
\let\sortvonname\swapthree
\end{verbatim}

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:

\begin{verbatim}
\printunsrtglossary[title={People (Alphabetical)},nonumberlist]
\printunsrtglossary[type={bybirth},target={false},nonumberlist]
\end{verbatim}

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:

\begin{verbatim}
\renewcommand*{\glolinkprefix}{idx:}
\end{verbatim}

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:

\begin{verbatim}
\printunsrtglossary*
[style={bookindex}]
\%
\let\sortname\swaptwo
\let\sortart\swaptwo
\let\sortvonname\swapthree
\renewcommand{\glsxtrgroupfield}\{indexgroup\}\
\renewcommand*{\glolinkprefix}\{idx:}\%
\end{verbatim}

The complete document code is listed below. The document build is:
pdflatex sample-people
bib2gls --group --break-space sample-people
pdflatex sample-people

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=altilistgroup]{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,
interpret-preamble=false
]
\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}
]
\GlsXtrLoadResources[
src={people},
type=index,
category=people,
action={define or copy},
copy-action-group-field={indexgroup},
selection={recorded and deps and see}
]
\newcommand\glsxtrpostlinkpeople{%
    \glsxtrifwasfirstuse{
        \ifglshasfield{user3}\glslabel\space(\glscurrentfieldvalue)\%
    }{}
}
\newcommand\glsxtrpostnamepeople{%
    \ifglshasfield{user3}\glscurrententrylabel\space(\glscurrentfieldvalue)\%
    {}
}
\newcommand\glsxtrpostdescpeople{%
    \ifglshasfield{user1}\glscurrententrylabel
    \space \ifglshasfield{user2}\glscurrententrylabel\space \glscurrentfieldvalue\,--\,\%
    \ifglshasfield{user2}\glscurrententrylabel\space \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[title={People (Alphabetical)},nonumberlist]
  \printunsrtglossary[type=bybirth,target=false,nonumberlist]
8 \textbf{Examples}: \texttt{sample-authors.tex}

\begin{verbatim}
\printunsrtglossary*[type=index,style=bookindex]
\%
  \let\sortname\swaptwo
  \let\sortart\swaptwo
  \let\sortvonname\swapthree
  \renewcommand{\glsxtrgroupfield}{indexgroup} %
  \renewcommand*{\glolinkprefix}{idx:}%
}
\end{document}

\texttt{sample-authors.tex}

This example uses the files \texttt{people.bib}, \texttt{books.bib}, \texttt{no-interpret-preamble.bib} and \texttt{interpret-preamble2.bib}. The aim is to reference the books in \texttt{books.bib} and have them listed by author. This means finding a way of assigning each book entry a \texttt{parent} field that contains the label identifying the relevant author in \texttt{people.bib}. I've used a style that shows letter group headings so I need to use the \texttt{--group} switch.

To recap, each author is defined in \texttt{people.bib} in the form:

@entry{dickens,
  name={\texttt{\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 \texttt{books.bib} in the form:

@entry{bleakhouse,
  name={Bleak House},
  description={novel by Charles Dickens},
  identifier={book},
  author={\texttt{\sortmediacreator{Charles}{Dickens}}},
  year={1852}
}

There's a field here (the custom \texttt{author} field) that contains the author's name, and this can be aliased to the \texttt{parent} field with \texttt{field-aliases}:

\texttt{field-aliases={author=parent}}

but the author's label in the \texttt{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 Macedonia (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).

Index
A
Alexanders III of Macedon, Alexander the Great, see Alexander III of Macedon.
B
Bonaparte, Napoleon, see Napoleon Bonaparte.
C
Caesar, Julius, see Gaius Julius Caesar.
V
Von Richthofen, Manfred, see Manfred von Richthofen.
W
Wellesley, Arthur, see Arthur Wellesley (1st Duke of Wellington).

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`:

```latex
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`:

```latex
field-aliases={identifier=category,author=parent}
```

so that the document can set the `textformat` and `glossnamefont` attributes:

```latex
\newcommand*{\bookfont}{\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 `\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:

```latex
check-end-punctuation={name}
```

With `glossaries-extra v1.23+` it’s easy to hook into the post-link hook to check if `nameendpunc` exists:

```latex
\renewcommand*{\glsxtrifcustomdiscardperiod}{%\GlsXtrIfFieldUndef{nameendpunc}{\glslabel}{#2}{#1}%
}
```

This will now cause the full stops following:

```latex\gls{whydidnttheyaskevans}.\gls{doandroidsdreamofelectricsheep}.\```

430
to be discarded.

The complete document code is listed below. The document build is:

```
pdflatex sample-authors
bib2gls --group sample-authors
pdflatex sample-authors
```

The resulting document is shown in figure 8.11.

```latex
\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]{\textbf{#1}}
\glssetcategoryattribute{book}{textformat}{bookfont}
\glssetcategoryattribute{book}{glossnamefont}{bookfont}

% requires glossaries-extra v1.23
\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{whydidn'ttheyaskevans}. \gls{icecoldinalex}. \gls{thehobbit}. \gls{thelordoftherings}. \gls{thewonderfulwizardofoz}. \gls{whiskygalore}.
```

431
\gls{whereeaglesdare}. \gls{icestationzebra}. \gls{ubik}. 
\gls{doandroidsdreamofelectricsheep}. \gls{thetroublewithharry}. 
\gls{brightonrock}.

\printunsrtglossary[title={Author and Book List}]
\end{document}

\textbf{sample-citations.tex}

This example uses the \texttt{BibTeX} file \texttt{citations.bib} to create a document that has both a bibliography created by \texttt{BibTeX} and glossaries created by \texttt{bib2gls} listing the authors and the titles. There are no glossary reference commands, such as \gls{gls}, but \texttt{bib2gls} can be run with \texttt{--cite-as-record} to treat the \texttt{\cite} commands (written to the \texttt{.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 \texttt{altlist} and \texttt{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 \texttt{glossary-tree} and \texttt{glossary-list} packages and patch the styles. A full stop is automatically placed after the descriptions with \texttt{postdot}.

\begin{verbatim}
\usepackage[record,% using 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}
\end{verbatim}

Next I need to create the glossaries for the list of authors and list of titles:

\begin{verbatim}
\newglossary*{contributors}{Authors}
\newglossary*{titles}{Titles}
\end{verbatim}

The simplest way of assigning the authors to the \texttt{contributors} glossary and the titles to the \texttt{titles} glossary is to use:

\begin{verbatim}
type={contributors}
\end{verbatim}

in the resource set and provide a modified version of \texttt{\bibglsnewbibtexentry} that assigns \texttt{type} after the options:

\begin{verbatim}
\newcommand{\bibglsnewbibtexentry}[4]{%
 \longnewglossaryentry*{#1}{name={#3},#2,type={titles}}{#4}%
}
\end{verbatim}
1 Sample


Author and Book List

<table>
<thead>
<tr>
<th>B</th>
<th>Lyman Frank Baum</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Raymond Chandler</td>
</tr>
<tr>
<td>D</td>
<td>Philip K. Dick</td>
</tr>
<tr>
<td>G</td>
<td>Henry Graham Green</td>
</tr>
<tr>
<td>H</td>
<td>Sammel Dashiel Hammett</td>
</tr>
<tr>
<td>L</td>
<td>Christopher Guy Landon</td>
</tr>
<tr>
<td>M</td>
<td>Compton Mackenzie</td>
</tr>
<tr>
<td>S</td>
<td>Jack Trevor Story</td>
</tr>
<tr>
<td>T</td>
<td>John Ronald Reuel Tolkien</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8.11: sample-authors.pdf
The standard BibTeX entry types need aliasing to \texttt{@bibtexentry}:

\begin{verbatim}
entry-type-aliases={\GlsXtrBibTeXEntryAliases}
\end{verbatim}

and the \texttt{title} field is aliased to \texttt{name}:

\begin{verbatim}
field-aliases={title=name}
\end{verbatim}

(The other fields aren’t required for the glossary lists.) The \texttt{category} is set to the original entry type:

\begin{verbatim}
category={same as original entry}
\end{verbatim}

So, for example, an entry that’s provided in the \texttt{.bib} file with \texttt{@article} has the category field set to \texttt{article}. (Compare this with \texttt{category={same as entry}} which would set the 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:

\begin{verbatim}
sort-field={category}
\end{verbatim}

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:

\begin{verbatim}
sort-suffix={name}
\end{verbatim}

Next the groups identified by the labels \texttt{article} and \texttt{book} are assigned titles.

\begin{verbatim}
\glsxtrsetgrouptitle{article}{Articles}
\glsxtrsetgrouptitle{book}{Books}
\end{verbatim}

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}:

\begin{verbatim}
\printunsrtglossary*[type={titles},style={indexgroup}]
{%
  \renewcommand{\glsxtrgroupfield}{category}\
  \renewcommand{\glstreenamefmt}[1]{\textbf{#1}}\
  \renewcommand{\glstreegroupheaderfmt}[1]{\textbf{#1}}%
}%
\end{verbatim}

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}}:
Note that this needs to be done for each \TeX 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}%
    {\contributorhandler}%
  }%
  {\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}\n\texttt{bib2gls --cite-as-record sample-citations}\n\texttt{bibtex sample-citations}\n\texttt{pdflatex sample-citations}\n\texttt{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:
stylen=\{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}[1]{\emph{#1}}}%


8 Examples: sample-msymbols.tex

\renewcommand{\glstreegroupheaderfmt}[1]{\textbf{#1}}%

\bibliographystyle{unsrt}
\bibliography{citations}
\end{document}

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

Figure 8.12: sample-citations.pdf

sample-msymbols.tex

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

437
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 location field:

```
save-locations={false},
selection={all}
```

Since I’m using a style that’s based on alttree I need to find the widest name, which can be done with 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 field-aliases to convert the custom identifier field to category:

```
field-aliases={identifier=category}
```

and sort by the category field:

```
sort-field={category}
```

Since this will cause identical sort values, I need to provide a way of ordering these identical values. Here I’ve decided to fallback on the description field:

```
identical-sort-action={description}
```

This means that entries will be ordered by category and then 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 sort-suffix as in the sample-textsymbols.tex file.

Remember that I want a small vertical gap between each logical block. These need the 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 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 group field to fetch its value from the category field:

```
replicate-fields={category=group}
```

(Since the field-aliases option is always performed before replicate-fields, the category field will already have been set and is available for replicating.)

This means that I’m essentially sorting by the 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 sample-textsymbols2.tex would end up with titled blocks, which I don’t want here. By using resource options such as field-aliases and replicate-fields I can avoid the warning that’s triggered with the default --warn-non-bib-fields.

The complete document code is listed below. The document build is:
8 Examples: sample-maths.tex

pdflatex sample-msymbols
bib2gls sample-msymbols
pdflatex sample-msymbols

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}

**sample-maths.tex**

This example uses bigmathsymbols.bib and mathsobjects.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}
## 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>ò</td>
<td>double integral</td>
</tr>
<tr>
<td>ñ</td>
<td>integral</td>
</tr>
<tr>
<td>Ï</td>
<td>surface integral</td>
</tr>
<tr>
<td>â</td>
<td>triple integral</td>
</tr>
<tr>
<td>Í</td>
<td>volume integral</td>
</tr>
<tr>
<td>ð</td>
<td>n-ary circled dot operator</td>
</tr>
<tr>
<td>ë</td>
<td>n-ary circled plus operator</td>
</tr>
<tr>
<td>Ï</td>
<td>n-ary circled times operator</td>
</tr>
<tr>
<td>Ë</td>
<td>n-ary coproduct</td>
</tr>
<tr>
<td>õ</td>
<td>n-ary intersection</td>
</tr>
<tr>
<td>ò</td>
<td>n-ary logical and</td>
</tr>
<tr>
<td>Ï</td>
<td>n-ary logical or</td>
</tr>
<tr>
<td>ë</td>
<td>n-ary product</td>
</tr>
<tr>
<td>Ï</td>
<td>n-ary square intersection operator</td>
</tr>
<tr>
<td>ë</td>
<td>n-ary union</td>
</tr>
<tr>
<td>Ï</td>
<td>n-ary union operator with plus</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 `format` to `user1` since `\glsxtrfmt` defaults to that field. If I decided to use a different field I also need to remember to redefine `\GlsXtrFmtField` to match.

As with `sample-msymbols.tex` I’m sorting by the `category` label and this value is copied to the `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 `selection` option I used in the `sample-msymbols.tex` example, however I still don’t want any number lists so I still have `save-locations={false}`.

I want `\glsxtrfmt` to index the term (which it doesn’t by default) so that means I need to redefine `\GlsXtrFmtDefaultOptions` to prevent it from using `noindex`:

```latex
\renewcommand{\GlsXtrFmtDefaultOptions}{}
```

I’ve provided some convenient wrapper commands that use `\glsxtrfmt*` or the non-linking `\glsxtreentryfmt` that are in the form:

```latex
\newcommand{\set}[2][]{{\glsxtrfmt*[#1]{set}{#2}}}
\newcommand{\nlset}[1]{{\glsxtreentryfmt{set}{#1}}}
```

The use of the starred form allows:

```latex
[\set{A} = \gls{bigcup}_{i=1}^n \set{B}_i] \]
```

which produces:

$$A = \bigcup_{i=1}^n B_i$$

Note the difference if the optional arguments aren’t used:

```latex
[\set{A} = \gls{bigcup}_{i=1}^n \set{B}_i] \]
```

This produces:

$$A = \bigcup_{i=1}^n B_i$$

Be careful with the set cardinality example. You might be tempted to nest `\set` within the argument of `\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 `\nlset` and `\nlsetcard`.

The complete document code is listed below. The document build is:

```bash
pdflatex sample-maths
bib2gls sample-maths
pdflatex sample-maths
```

The resulting document is shown in figure 8.14.
8 Examples: sample-maths.tex

\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{amssymb}
\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={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*[#1]{set}{#2}}
\newcommand{\nlset}[1]{\glsxtrentryfmt{set}{#1}}
\newcommand{\setq}[2][]{\glsxtrfmt*[#1]{setcontents}{#2}}
\newcommand{\nlsetq}[1]{\glsxtrentryfmt{setcontents}{#1}}
\newcommand{\setmember}[2][]{\glsxtrfmt*[#1]{setmembership}{#2}}
\newcommand{\nlsetmember}[1]{\glsxtrentryfmt{setmembership}{#1}}
\newcommand{\setcard}[2][]{\glsxtrfmt*[#1]{setcard}{#2}}
\newcommand{\nlsetcard}[1]{\glsxtrentryfmt{setcard}{#1}}
\newcommand{\transpose}[2][]{\glsxtrfmt*[#1]{transpose}{#2}}
\newcommand{\nltranspose}[1]{\glsxtrentryfmt{transpose}{#1}}
\newcommand{\inv}[2][]{\glsxtrfmt*[#1]{inverse}{#2}}
\newcommand{\nlinv}[1]{\glsxtrentryfmt{inverse}{#1}}
\newcommand{\vtr}[2][]{\glsxtrfmt*[#1]{vector}{#2}}
\newcommand{\nlvtr}[1]{\glsxtrentryfmt{vector}{#1}}
\newcommand{\mtx}[2][]{\glsxtrfmt*[#1]{matrix}{#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:
\[
\set{B}_1 = \setcontents{1, 3, 5, 7}, \quad \set{B}_2 = \setcontents{2, 4, 6, 8}, \quad \set{B}_3 = \setcontents{9, 10} \\
\]
Define:
\[
\set{A} = \gls{bigcup}_{i=1}^3 \set{B}_i = \setcontents{1, \ldots, 10} \\
\]
The cardinality of a set $\gls{set}$ is denoted $\gls{setcard}$ and is the number of elements in the set.
\[
\setcard{\set{B}_1} = 4, \quad \setcard{\set{B}_2} = 4, \quad \setcard{\set{B}_3} = 2, \quad \setcard{\set{B}_1 \cup \set{B}_2} = 8 \quad \setcard{\gls{emptyset}} = 0 \\
\]

\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).
\[
\gls{identitymatrix}\Vtr{x} = \Vtr{x}, \quad \Mtx{A}\inv{\Mtx{A}} = \gls{identitymatrix}, \quad \inv{\nVtr{x}}\gls{1vec} = \gls{sum}_{i} x_i \\
\]
\printunsrtglossaries

443
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 \texttt{\textbackslash{}Sun} and \texttt{\textbackslash{}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}
\undefined{Sun}
\undefined{Lightning}
\usepackage[weather]{ifsym}

The etoolbox package is also loaded as it provides \texttt{\textbackslash{}undefined}. (An alternative is to modify the miscsymbols.bib file so that it uses ifsym’s more generic \texttt{\textbackslash{}textweathersymbol} command and omit the \texttt{\textbackslash{}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 \texttt{\textbackslash{}identical-sort-action}, I’m using \texttt{\textbackslash{}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 $\mathbb{S}$) is characterised by a set of entities with a set of axioms. For example:

$$\mathbb{N} = \{x : x \text{ is a positive integer}\}$$
$$\mathbb{Z} = \{x : x \text{ is an integer}\}$$
$$\mathbb{R} = \{x : x \text{ is a real number}\}$$

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).

$$Ix = x, \quad AA^{-1} = I, \quad x^{-1} = \sum_i x_i$$

Glossary

- $I$ the identity matrix.
- $M^{-1}$ the inverse of $M$.
- $M$ a matrix.
- $v$ a vector.
- $I$ the vector of 1s.
- $\sum\sum$ $n$-ary summation.
- $\bigcup\bigcup$ $n$-ary union.
- $\mathbb{S}$ a number space.
- $|\mathbb{S}|$ the cardinality of $\mathbb{S}$.
- $\emptyset$ the empty set.
- $\{\ldots\}$ set contents.
- $\{x : \ldots\}$ set membership.
- $U$ the universal set.
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:

\texttt{set-widest}

In this case, \texttt{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 \texttt{glossaries-extra-stylemods} package.

This is actually not the best method as \texttt{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 \texttt{sample-textsymbols2.tex}.

The complete document code is listed below. The document build is:

\texttt{pdflatex sample-textsymbols}
\texttt{bib2gls sample-textsymbols}
\texttt{pdflatex sample-textsymbols}

The resulting document is shown in figure 8.15.

\begin{verbatim}
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{etoolbox}
\usepackage{marvosym}
\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={

8 Examples: \texttt{sample-textsymbols2.tex}

\begin{verbatim}
\begin{document}
\printunsrtglossaries
\end{document}
\end{verbatim}

\texttt{sample-textsymbols2.tex}

This example is a better approach than the \texttt{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 \texttt{@icon} must be aliased for the entries to be recognised:

\begin{verbatim}
entry-type-aliases={icon=symbol}
\end{verbatim}

The \texttt{topics.bib} file contains terms with labels that match the custom \texttt{identifier} fields used in the \texttt{miscsymbols.bib} file. So both files are loaded and the \texttt{identifier} field is now aliased to \texttt{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 \texttt{name} field) instead of by the label.

\begin{verbatim}
src={topics,miscsymbols},
field-aliases={
  identifier=parent,
  icon=name,
  icondescription=description},
\end{verbatim}

There's no \texttt{sort-field} option in this example. The default \texttt{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 (\texttt{@index} and \texttt{@indexplural}), I want to sort by the \texttt{name}, which is the default fallback if the \texttt{sort} field is missing for the index entry types.

The default fallback for the \texttt{sort} field for \texttt{@symbol} entries is the label, but in this case I want to use the \texttt{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.
symbol-sort-fallback={description}

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, stylemods={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.

\documentclass[a4paper]{article}

\usepackage[T1]{fontenc}
\usepackage{etoolbox}
\usepackage{marvosym}

% package conflict, need to undefine conflicting commands
\textbackslash undef\textbackslash Sun
\textbackslash undef\textbackslash Lightning
\textbackslash undef\textbackslash ifsym

\usepackage[weather]{ifsym}

\usepackage[record, % using bib2gls
nostyles, % don’t load default styles
postdot, % append a dot after descriptions
stylemods={tree,topic}, % load glossary-tree.sty and glossary-topic.sty
style=topic]{glossaries-extra}

\GlsXtrLoadResources[
src={topics,miscsymbols},
% make @icon behave like @symbol:
entry-type-aliases={icon=symbol},

449
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
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 `@preamble` definition, which is now in the resource file.

The tagging format is governed by \glsxtrtagfont which underlines its argument by default. I’ve redefined it to also convert the letter to upper case:

\renewcommand*{\glsxtrtagfont}[1]{\underline{\MakeTextUppercase{#1}}}

Note that in the `mathml` case, the first tag consists of more than one letter:

long={\abbrvtag{m\NoCaseChange{ath}}ematical }#markuplang

Here \NoCaseChange prevents \MakeTextUppercase 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 \glsxtrshort, which 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 `glsignore` or suppress the indexing by changing the default options with \GlsXtrSetDefaultGlsOpts. In this case I decided to turn the records into ignored records:

\GlsXtrSetDefaultNumberFormat{glsignore}

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:

\newcommand{\glsxtrpostdescmarkuplanguage}{%\glsxtrifhasfield{location}{\glscurrententrylabel}{,}{.}}

I’ve used `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 glossname attribute rather than using `name-case-change`:

\glssetcategoryattribute{markuplanguage}{glossname}{firstuc}

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 `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 `bib2gls` instead:

description-case-change={firstuc}
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:
stylemods={tree},
style=treegroup\]{glossaries-extra}
\setabbreviationstyle[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}[1]{\underline{\MakeTextUppercase{#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}

sample-usergroups.tex

This example uses usergroups.bib. This requires \LaTeX{} or \Lualatex{} as the .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 \emph{name} or \emph{long} field, it also includes the custom field \emph{translation} that provides an (approximate) translation. If this field is present, the language is given by the first element of the custom \emph{language} field.

In this case, I’m providing keys for the custom \emph{language} and \emph{translation} fields, and, for a bit of variety from the other examples, I’m ignoring the custom \emph{identifier} field. The custom keys are provided with \glsaddstoragekey:

\glsaddstoragekey{language}{}{\glsentrylanguage}
\glsaddstoragekey{translation}{}{\glsentrytranslation}

The .bib file includes abbreviations. Remember that the abbreviation style must be set before the resource file is loaded:

\setabbreviationstyle[tug]{long-short-user}

For this example, I’m explicitly setting the \emph{category} field to tug:

\emph{category}={tug}

Some of the fields end with a full stop. This isn’t a problem with the \emph{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 \emph{short} field ends with a full stop. This means I need to check for end-of-sentence punctuation for the \emph{short} field. It’s also a good idea to do this for the \emph{name} field for the non-abbreviations.

\emph{check-end-punctuation}={name,short}

454
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 \TeX 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.

Figure 8.17: sample-markuplanguages.pdf
It's now possible to discard a full stop that follows `\gls`:

\renewcommand*{\glsxtrifcustomdiscardperiod}[2]{% 
  ifglshashshort{\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 `XƎL3TEX` or `LuaL3TEX` 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 `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 `ietf` language tag to a language name recognised by `polyglossia`.

Iterating over all entries can be done with `\forglsentries` but remember that no entries will be defined before `bib2gls` has been run, so this does nothing on the first \LaTeX run.
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 \glsxtr forcsvfield which iterates over all elements of the given field (language in this case) and break out of the loop (with \glsxtrendfor) 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 \glsxtr forcsvfield 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}%}
Once the \forcglsentries 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}{\textit{\textbf{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}\\
     \ifglishasfield{\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 \textit{description} field, so I’ve provided a post-name hook to append it in parentheses (with the translation, if provided):

\newcommand{\glsxtrpostnametug}{
\ifglshasdesc{\glscurrententrylabel}{
\if\glscurrentfieldvalue{translation}
{\textenglish{\glscurrentfieldvalue}}%
{%
}\
\else
{\textenglish{\glscurrentfieldvalue}}%
{%
}}%
\else
{%
\if\glscurrentfieldvalue{translation}
{\textenglish{\glscurrentfieldvalue}}%
{%
}\
\else
{}%
}
}

Remember that this hook is included within the \textit{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:

\texttt{xelatex sample-usergroups}
\texttt{bib2gls --group sample-usergroups}
\texttt{xelatex sample-usergroups}
\texttt{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}
\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]{%
  \ifglsishasshort{\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.1df}%
  {%}
  {\xifinlist{\thislanguage}{\langlist}{}}%
  {\listxadd{\langlist}{\thislanguage}}%
  \xGlsXtrSetField{\thislabel}{useri}{text\thislanguage}%
  \glsxtrendfor
\glssetcategoryattribute{tug}{textformat}{tugtextformat}
\glssetcategoryattribute{tug}{glossnamefont}{tugnameformat}

\newcommand{\glsxtrpostnametug}{%
  \ifglshasdesc{\glscurrententrylabel}{%
    (\glossentrydesc{\glscurrententrylabel}%
    \glsxtrifhasfield{translation}{\glscurrententrylabel}{%
      , \textenglish{\glscurrentfieldvalue}}%
    )%
  }%
}{%
  \glsxtrifhasfield{translation}{\glscurrententrylabel}{%
    \glsxtrparen{#1%
    \ifglshasfield{\glsxtruserfield}{#2}{, \textenglish{\glscurrentfieldvalue}}%}
  }%
}
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{cvt}. \gls{MaTeX}. \gls{ITALIC}.
\gls{IsTeX}. \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{cvt}. \gls{MaTeX}. \gls{ITALIC}.
\gls{IsTeX}. \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:

\texttt{selection={recorded and deps and see}}

to ensure the aliased entries are selected.
8 Examples: sample-multil1.tex

1 Sample

1.1 First Use

1.2 Next Use
bgTeX (Bulgarian LATEX Users Group), CervanTeX (Grupo de Usuarios de TEX Hispanohablantes, Spanish Speaking TEX Users Group), CSTUG (Československé sdružení uživatelů TeXu, z. s., Czech Republic TEX Users Group), CTeX (Chinese TEX Society), DANTE e.V. (Deutschsprachige Anwendervereinigung TEX e.V., German Speaking TEX Users Group), DK-TUG (Danish TEX Users Group), Estonian User Group, CervanTeX (Grupo de Usuarios de TEX Hispanohablantes, Spanish Speaking TEX Users Group), Tirant lo TEX, GUTenberg (Groupe francophone des utilisateurs de TEX, French Speaking TEX Users Group), UK-TUG (UK TEX Users Group), ɛϕτ (Σύλλογος Ελλήνων Φίλων του TEX, Greek TEX Friends), MaTeX (Magyar TEX Egyesület, Hungarian TEX Users Group), ITALIC (Irish TEX and LATEX In-print Community), ÍsTeX (Vefur íslenskra TEX notenda, Icelandic TEX Users Group), GuIT (Gruppo Utilizzatori Italiani di TEX, Italian TEX Users Group), KTS (Korean TEX Society), latex-br (Grupo de Usuários, Brazilian TEX Users Group), Lietuvos TEX'o Vartotojų Grupė (Lithuanian TEX Users Group), TEX México, NTG (Nederlandstalige TEX Gebruikersgroep, Netherlands TEX Users Group), Nordic TEX Users Group, GUST (Polska Grupa Użytkowników Systemu TEX, Polish TEX Users Group), GUTpt (Grupo de Utilizadores de TEX, Portuguese TEX Users Group), VietTUG (Vietnamese TEX Users Group), LUGSA (LATEX User Group — South Africa).

Glossary

B
bgTeX (Bulgarian LATEX Users Group), 1
C
CervanTeX (Grupo de Usuarios de TEX Hispanohablantes, Spanish Speaking TEX Users Group), 1
CSTUG (Československé sdružení uživatelů TeXu, z. s., Czech Republic TEX Users Group), 1
CTeX (Chinese TEX Society), 1
D
DANTE e.V. (Deutschsprachige Anwendervereinigung TEX e.V., German Speaking TEX Users Group), 1
DK-TUG (Danish TEX Users Group), 1
E
Estonian User Group, 1
G
GuIT (Gruppo Utilizzatori Italiani di TEX, Italian TEX Users Group), 1
GUST (Polska Grupa Użytkowników Systemu TEX, Polish TEX Users Group), 1
GUTenberg (Groupe francophone des utilisateurs de TEX, French Speaking TEX Users Group), 1
GUTpt (Grupo de Utilizadores de TEX, Portuguese TEX Users Group), 1
I
ÍsTeX (Vefur íslenskra TEX notenda, Icelandic TEX Users Group), 1
ITALIC (Irish TEX and LATEX In-print Community), 1
K
KTS (Korean TEX Society), 1
L
latex-br (Grupo de Usuários, Brazilian TEX Users Group), 1
Lietuvos TEX'o Vartotojų Grupė (Lithuanian TEX Users Group), 1
LUGSA (LATEX User Group — South Africa), 1
M
MaTeX (Magyar TEX Egyesület, Hungarian TEX Users Group), 1
N
Nordic TEX Users Group, 1
NTG (Nederlandstalige TEX Gebruikersgroep, Netherlands TEX Users Group), 1
T
TEX México (Mexican TEX Users Group), 1
Tirant lo TEX (Catalan TEX Users Group), 1
TUG (TEX Users Group), 1
U
UK-TUG (UK TEX Users Group), 1
V
VietTUG (Vietnamese TEX Users Group), 1
Ɛ
ɛϕτ (Σύλλογος Ελλήνων Φίλων του TEX, Greek TEX Friends), 1

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}
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 \texttt{bib2gls} treat them as though they were, and I also need to alias the custom @chemical, @unit and @measurement entry types:
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,
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 \glstreeheaderfmt to prevent the names appearing in bold, which means I also need to redefine \glsentryheaderfmt 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}
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 “bílinite” 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 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`:

```latex
\texttt{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`:

```latex
\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%
  \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}
}%
```

However the chemical formulae look a little odd in parentheses (especially those that contain parenthetical parts) but this can be fixed by adding a category check:

```latex
\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%
  \ifglshassymbol{#1}%
    {%
      \glsifcategory{#1}{chemical}%
      {, \glossentrysymbol{#1}%
        {\space(\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:

```latex
\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%
  \ifglshassymbol{#1}%
    {%
      \glsifcategory{#1}{chemical}%
      {, \glossentrysymbol{#1}%
        {\space(\glossentrysymbol{#1})}%
        }
    }%
  {}%
}%
```

468
However, since I decided not to store the short form in the symbol field and just saved the dual entry label instead, I need to lookup the short form from the dual entry:

\renewcommand*{\glsxtrbookindexname}[1]{\glossentryname{#1}%
\ifglshassymbol{#1}%
{\% \glsifcategory{#1}{chemical}\%
, \glossentrysymbol{#1}\%
{\% \space(\glossentrynameother{#1}{symbol})\%
}\%}
{% \glsifcategory{#1}{markuplanguage}\%
{\% \glsxtrifhasfield{short}{\glsxtrusefield{#1}{dual}}\%
\space(\glscurrentfieldvalue)\%
{\}%
{\}%
{\}%
}}

Not all of the markup languages are abbreviations so this uses \glsxtrifhasfield to check if the short field is set. The dual entry’s label is easily obtained because dual-field has provided the 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 \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 \LaTeX and 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
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
nostreams, % don't load default styles
% load and patch required style packages:
{style.mods = {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}
\setabbreviationstyle[bacteria]{long-only-short-only}
\setabbreviationstyle[markuplanguage]{long-short-desc}
\renewcommand*{\glsxtrlongshortdesctoken}{% 
  \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,
8 Examples: sample-muti1.tex

label-prefix={idx.},
dual-prefix={},
cs-label-prefix={},
combine-dual-locations={primary},
dual-field,
sort={en},
sort-field={name},
dual-type={same as category},
dual-sort={en},
dual-sort-number-pad={2},
dual-sort-pad-plus={1},
dual-sort-pad-minus={0},
dual-break-at=upper-upper-word,
entry-type-aliases={
  abbreviation=dualindexabbreviation,
  entry=dualindexentry,
  symbol=dualindexsymbol,
  unit=dualindexsymbol,
  measurement=dualindexsymbol,
  chemical=dualindexsymbol
},
abbreviation-name-fallback={long},
symbol-sort-fallback={name},
field-aliases={
  identifier=category,
  formula=symbol,
  chemicalname=name,
  unitname=name,
  unitsymbol=symbol,
  measurement=description
},
\newcommand{\bacteriafont}[1]{\emph{#1}}
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont}
\glssetcategoryattribute{markuplanguage}{glossdesc}{firstuc}
\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname[#1]%
  \ifglsifcategory{\#1}{chemical}%
  {, \glossentrysymbol[#1]}%
}
8 Examples: sample-multi1.tex

\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}
472
\section{Chemicals}
\gls{duck}, \gls{parrot}, \gls{hedgehog}, \gls{sealion}.
\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{Zr3P044}, \gls{ZnF2}.

\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]{\glsadd[format=hyperbf]{#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]{\%}\renewcommand*{\glstreegroupheaderfmt}[1]{\textbf{#1}}{{\%}}}\printunsrtglossary[type=baseunit,style=alttree]
\printunsrtglossary[type=derivedunit,style=alttree]
\end{document}}

\textbf{sample-multi2.tex}

This example is an alternative approach to \texttt{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, Bacillus subtilis, Clostridium tetani, Pseudomonas aeruginosa, Pseudomonas fluorescens, Clostridium tetani, Bacteroides melanos, Rickettsia rickettsii.

1.1.2 Next Use
C. botulinum, P. putida, B. subtilis, C. tetani, P. aeruginosa, P. fluorescens, B. melanos, R. rickettsii, E. coli.

1.2 Markup Languages

1.2.1 First Use
XML, markup language, a format of HTML designed to separate content from style.

1.2.2 Next Use
XML, markup language, a format of HTML designed to separate content from style.

1.3 Vegetables
Broccoli, artichoke, spinach, green, leafy vegetable, immature fruit of a vegetable marrow.

1.4 Minerals
Beryl, amethyst, chalcedony, aquamarine, aragonite, bílinite, cyanotrichite, a hydrous copper aluminium sulfate mineral.

1.5 Animals
Duck, parrot, hedgehog, sea lion.

1.6 Chemicals
Al2(SO4)3, H2O, C6H12O6, CH3CH2OH, CH2O, OF2, O2F2, SO42–, H3O+, OH–, O2, H2O, AlF3, O, Al2CoO4, As4S4, as a complex of copper carbonate, a crystal form of calcium carbonate, anhydrous carbonate mineral.

1.7 SI Units
Base: A, kg, m, s, K, mol, cd, cm, m, m2, m3, kg/m, kg m−1, m2 kg−1, m−1.
Derived: m2, m3, m s−1, m s−2, A m−2, cd m−2, m3 kg−1, mol m−3, m−1.

Glossaries

Bacteria
B. subtilis, Bacillus subtilis,
C. tetani, Clostridium tetani,
P. composti, Planifilum composti,
P. fimeticola, Planifilum fimeticola.

Markup Languages
HTML (hypertext markup language), The standard markup language for creating web pages.

XHTML (extensible hypertext markup language), A lightweight markup language with plain text formatting syntax.

MathML (mathematical markup language), A standard markup language for describing mathematical notation.

XML (extensible markup language), A markup language that defines a set of rules for encoding documents.

Vegetables
Brussels sprout, artichoke, spinach, green, leafy vegetable, immature fruit of a vegetable marrow.

Minerals
A, a crystal form of calcium carbonate, a crystal form of calcium carbonate, a crystal form of calcium carbonate, a crystal form of calcium carbonate.
Animals

A hare is a long-eared rodent with a thick coat.

A bluebird is a small bird with a blue plumage.

A seal is a large type of seal.

Chemical Formula

- Al₂O₃: aluminium oxide
- AlF₃: aluminium trifluoride
- CO₂: carbon dioxide
- COOH: carbonyl group
- CN⁻: carbonyl anion

SI Units

- kg: kilogram
- m: metre
- s: second
- mol: mole
- A: ampere
- cd: candela
- m⁻¹: metre per metre
- m⁻²: metre per square metre
- m⁻³: metre per cubic metre
- s⁻¹: second per second
- mol m⁻³: mole per cubic metre
- A m⁻²: ampere per square metre

Derived Units

- W: watt
- Pa: pascal
- kg m⁻¹ s⁻²: kilogram per metre per second squared
- J: joule
- N: newton
- L: litre

Index

A: aluminium oxide, Al₂O₃, 2, 4
B: benzene, C₆H₆, 3
C: carbon dioxide, CO₂, 2
D: dioxygen, O₂, 2
E: ethanol, CH₃OH, 2
F: formaldehyde, CH₂O, 2

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 \texttt{BibX} 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 \texttt{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.
As with the sample-people.tex document, I need to use the \texttt{--break-space} switch to convert the ~ to a normal breakable space so that it matches the given format. I’ve loaded the datetime2 package:

\usepackage[en-GB]{datetime2}

so that I can use \texttt{\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 \texttt{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 \texttt{category=people}, this document obtains the \texttt{category} field from the custom \texttt{identifier} field, which in this case has the value \texttt{person}. This means that the category hooks from sample-people.tex need to be renamed to reflect the different category label:

\newcommand*{\glsxtrpostlinkperson}{\glsxtrifwasfirstuse}{%\ifglshasfield{user3}{\glslabel}{\space{\glscurrentfieldvalue}}}%

\newcommand*{\glsxtrpostnameperson}{%\ifglshasfield{user3}{\glscurrententrylabel}{\space{\glscurrentfieldvalue}}}%

\footnote{The \texttt{en-GB} option to datetime2 also requires that datetime2-english must be installed.}
\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:
8 Examples: sample-multi2.tex

\newcommand{\bookfont}[1]{\textit{#1}}
\glssetcategoryattribute{book}{textformat}{bookfont}
\glssetcategoryattribute{book}{glossnamefont}{bookfont}

(as for sample-media.tex) and for films:

\newcommand{\filmfont}[1]{\textit{#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 name field is obtained from the custom measurement field. Since this contains a word, the default locale sort is appropriate. I’ve locally redefined \glsxtralttreeSymbol-DescLocation to place the symbol in parentheses after the description:
The base units are replicated in the baseunit glossary, this time with the \texttt{name} field obtained from the custom \texttt{unitsymbol} field. This means that I need to find a way to prevent duplicate labels. The simplest method is to use \texttt{duplicate-label-suffix}:

\begin{verbatim}
\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}
]
\end{verbatim}

I can’t use \texttt{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 \texttt{\glsFindWidestTopLevelName} to find it:

\begin{verbatim}
\printunsrtglossary*[type={baseunit},style={alttree},nogroupskip]
{\% \\
  \glsFindWidestTopLevelName[baseunit]{}}
\end{verbatim}

The text symbols from \texttt{missymbols.bib} are all loaded in a single resource set, where the \texttt{type} field can be obtained from the \texttt{category}, which in turns is obtained from the custom \texttt{identifier} field. Since \texttt{bib2gls} doesn’t recognise any of the symbol commands, I’m sorting according to the \texttt{description} field. (Even if \texttt{bib2gls} could determine a Unicode value for each of the symbols, sorting by the description makes more sense in this case.)

\begin{verbatim}
\GlsXtrLoadResources[
  src={missymbols},
  field-aliases={
    identifier=category,
    description=category
  },
  category={measurement},
  type={text},
  symbol-sort-fallback={name},
  save-locations={false}
]
\end{verbatim}
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:

```latex
\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 \glssymbols\glsname.

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)):

```latex
\renewcommand*{\glsxtrlongshortdescname}{%
    \protect\glslongfont{\the\glslongtok}\space
    \glsxtrparen{\glsabbrvfont{\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 \texttt{@index} with the \texttt{name} field set to a word form (chemical name and icon description). This means they’re appropriate for alphabetical sorting. (Both \texttt{@entry} and \texttt{@symbol} require the \texttt{description} field, which is why I’ve aliased \texttt{@chemical} and \texttt{@icon} to \texttt{@index} here.) The custom \texttt{@measurement} entry type has a \texttt{description} field (obtained from \texttt{unitname}), so that’s aliased to \texttt{@entry} as again the \texttt{name} field is suitable for alphabetical sorting.

I’ve aliased \texttt{@unit} to \texttt{@dualentry} rather than \texttt{@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 \texttt{index} glossary:

\texttt{type={index}},
\texttt{dual-type={index}}

All .\texttt{bib} files used in the previous resource sets are needed as well as the \texttt{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:

\texttt{selection[recorded and deps and see]}

Again it’s necessary to provide a way to avoid duplicate entry labels, which can be done with

\texttt{duplicate-label-suffix={.copy}},

as above. However, this will cause the cross-references (from the \texttt{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:

\texttt{label-prefix={idx.}},
\texttt{record-label-prefix={idx.}},

This means that the entries defined in \texttt{terms.bib} need to be referenced with this prefix.

All instances of \texttt{\gls} will link to the original entry, so all entries except for those in the \texttt{terms.bib} file will link to the relevant glossary. Those in the \texttt{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 \texttt{category} to resolve identical sort values:

\texttt{identical-sort-action={category}}

For the people who have a \texttt{first} field, I’ve decided that this would be more appropriate for the index as it’s more compact than the \texttt{name}, so here I’ve done the reverse to earlier and copied the \texttt{first} field (if supplied) into the \texttt{name} field, but since the \texttt{name} field is already provided the override setting needs to be on:

\texttt{replicate-override},
\texttt{replicate-fields={first=name}}

As with \texttt{sample-people.tex} I’ve provided some custom commands to make it easier to locally redefine \texttt{\sortname} and \texttt{\sortvonname}:

\texttt{\newcommand*{\swaptwo}[2]{#2, #1}}
\texttt{\newcommand*{\swapthree}[3]{#2 #3, #1}}

I’ve redefined \texttt{\glsxtrbookindexname} in a similar manner to \texttt{sample-multi1.tex} but it has some modifications:
\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%
  \ifglsishassymbol{#1}%, \glossentrysymbol{#1}%
  \space(\glossentrynameother{#1}{symbol})%
}{%}
\glsifcategory{#1}{chemical}{%, \glossentrysymbol{#1}%
  \space(\glossentrynameother{#1}{symbol})%
}{%}
\glsifcategory{#1}{film}{%}
\_\_\_\_\_\_\_\_(film){%}
\}%
\}%
\}

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:
\newcommand*{\glsextrapostnamehook}{% 
\glsadd[format={hyperbf}]{#1}%
}\}

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:

\texttt{pdflatex sample-multi2} \texttt{bib2gls --group --break-space sample-multi2} \texttt{pdflatex sample-multi2} \texttt{bib2gls --group --break-space sample-multi2} \texttt{pdflatex sample-multi2}

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 \texttt{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={

unitname=description,
unitsymbol=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}%
}%
\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},
}
\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]{\textit{#1}}
\glssetcategoryattribute{bacteria}{textformat}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossnamefont}{bacteriafont}
\glssetcategoryattribute{bacteria}{glossdescfont}{bacteriafont}
\newcommand{\bookfont}[1]{\textit{#1}}
\glssetcategoryattribute{book}{textformat}{bookfont}
\glssetcategoryattribute{book}{glossnamefont}{bookfont}
8 Examples: sample-multi2.tex

\newcommand{\filmfont}[1]{\emph{#1}}
\glssetcategoryattribute{film}{textformat}{filmfont}
\glssetcategoryattribute{film}{glossnamefont}{filmfont}
\glssetcategoryattribute{film}{glossdesc}{firstuc}
\glssetcategoryattribute{markuplanguage}{glossdesc}{firstuc}

\newcommand*{\glsxtrpostlinkmediacontrol}{\%
  \glsxtrpostlinkAddDescOnFirstUse
}
\newcommand*{\glsxtrpostlinkinformation}{\%
  \glsxtrpostlinkAddDescOnFirstUse
}
\newcommand*{\glsxtrpostlinkweather}{\%
  \glsxtrpostlinkAddDescOnFirstUse
}
\newcommand*{\glsxtrpostlinkperson}{\%
  \glxtrifwasfirstuse
  \%
    \ifglshasfield{user3}{\glslabel}{\%
        \space{\glscurrentfieldvalue}{\%
      \}
    \%
  \}
}
\newcommand*{\glsxtrpostnameperson}{\%
  \ifglshasfield{user3}{\glscurrententrylabel}{\space{\glscurrentfieldvalue}{\%
}
}
\newcommand*{\glsxtrpostdescperson}{\%
  \ifglshasfield{user1}{\glscurrententrylabel}{\%
    \space{\glscurrentfieldvalue}{\%
      \ifglshasfield{user2}{\glscurrententrylabel}{\%
        \space{\glscurrentfieldvalue}{\%
          \%
        \}
      \%
    \%
  \}
\%

490
8 Examples: sample-multi2.tex

\newcommand*{\glsxtrpostdescfilm}{%
  \ifglsishasfield{user1}{\glscurrententrylabel}%
    \glsxtrrestorepostpunc \requires\glossaries-extra v1.23+
    \ (released \glscurrentfieldvalue)%
  \}%
  \ifglsishasfield{user2}{\glscurrententrylabel}%
    \glsxtrrestorepostpunc
    featuring \glscurrentfieldvalue
  \}%
\}

\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%
  \ifglsishassymbol{#1}%
    \glsifcategory{#1}{chemical}%
    \, \glossentrysymbol{#1}%
    \space(\glossentrynameother{#1}{symbol})%
  \}%
  \glsifcategory{#1}{film}%
  \ (film)%
  \}%
\%
\%
\}

% requires glossaries-extra v1.25+:
\renewcommand*{\glsextrapostnamehook}[1]{%
  \glsadd[format=hyperbf]{#1}%
}\}
\newcommand{\unit}{\glssymbol}
\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{rricketttsii}.

\subsection{Next Use}
\gls{cbotulinum}, \gls{pputida}, \gls{cperfringens},
\gls{bsubtilis}, \gls{ctetani}, \gls{pcomposti},
\gls{pfimeticola}, \gls{cburnetii}, \gls{raustralis},
\gls{rricketttsii}.

\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{sealion},
\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}).

\section{Miscellaneous Symbols}

\subsection{First Use}
Idxpl{mediacontrol}: \gls{forward}, \gls{forwardtoindex}, \gls{rewind}, \gls{rewindtoindex}.

Idx{information}: \gls{bicycle}, \gls{coffeecup}, \gls{info}, \gls{gentsroom}, \gls{ladiesroom}, \gls{wheelchair}, \gls{football}, \gls{recycling}.
8 Examples: sample-multi2.tex

\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}{%\ifglshassymbol{#1}{space(\glossentrysymbol{#1})}{}%\glspostdescription\glsxtrAltTreePar} %}
\printunsrtglossary*[type=baseunit,style=alttree,nogroupskip]
{\percent}
8 Examples: sample-multi2.tex

\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}
1 Sample

1.1 Bacteria

This section is about bacteria.

1.1.1 First Use

Christobacter bactidea, Planosolomon princeps, (aka plasmid), Bacillus subtilis, Christobacter cover, Planosolomon conjunct, Planosolomon juctlandia, Florida hitter, Red

1.1.2 Next Use


1.2 Markup Languages

This section is about markup languages.

1.2.1 First Use

TEX, markdown, XHTML, MathML, SVG.

1.2.2 Next Use

Back to start of track, next track, (football stadium), (football stadium), (next track), (football stadium), (back to start of track).

1.3 Vegetables

This section is about vegetables.

1.3.1 First Use

Spinach, gette, spinach, gette, spinach, gette, spinach, gette, spinach, gette.

1.3.2 Next Use

Back to start of track.

1.4 Minerals

This section is about minerals.

1.4.1 First Use

Calcite, bílinite, cyanotrichite, biotite, dolomite, quetzalcoatlite, vulcanite.

1.4.2 Next Use

Back to start of track, next track, (football stadium), (football stadium), (next track), (football stadium), (back to start of track).

1.5 Animals

This section is about animals.

1.5.1 First Use

Duck, parrot, hedgehog, zander, aardvark, zebra, swan, armadillo.

1.5.2 Next Use

Duck, parrot, hedgehog, zander, aardvark, zebra, swan, armadillo.

1.6 Chemicals

This section is about chemicals.

1.6.1 First Use

HCl, NaOH, KOH (alkali), NH₄OH (ammonia), CH₃COOH (acetic acid), CO₂, carbon dioxide, CO₂, carbon dioxide, CO₂, carbon dioxide, CO₂, carbon dioxide.

1.6.2 Next Use

CO₂, carbon dioxide, CO₂, carbon dioxide.

1.7 SI Units

This section is about SI units.

1.7.1 First Use

Base SI units: m, s, kg, mol, cd, A, kelvin, cd m⁻², mol m⁻³, kg m⁻¹ s⁻¹.

1.7.2 Next Use

Back to start of track.

1.8 Books and Films

This section is about books and films.

1.8.1 First Use

Books: A Tale of Two Cities (by Charles Dickens), The Big Sleep, the Romantic Comedy, Why Didn't They Ask Evans? (by Agatha Christie (Lady

1.8.2 Next Use

Books: A Tale of Two Cities (by Charles Dickens), The Big Sleep, the Romantic Comedy, Why Didn't They Ask Evans? (by Agatha Christie (Lady

1.9 Miscellaneous Symbols

This section is about miscellaneous symbols.

1.9.1 First Use

Symbols: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

1.9.2 Next Use

Back to start of track.

2 Glossaries

2.1 Bacteria

This section is about bacteria.

2.1.1 First Use

Bacteria:

- Bacillus subtilis
- Clostridium tetani
- Clostridium botulinum
- Coxiella burnetii
- Planifilum composti
- Planifilum fimeticola
- Pseudomonas putida

2.1.2 Next Use

Back to start of track, next track, (football stadium).

5 Examples: sample-multi2.txt

Figure 8.21: sample-multi2.pdf (pages 1 to 4)
Animals

<table>
<thead>
<tr>
<th>Mammals</th>
<th>Terrestrial</th>
<th>Aquatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humans</td>
<td>8 Examples: sample-multi2.pdf (pages 5 to 8)</td>
<td>2 Examples: sample-multi2.pdf (pages 5 to 8)</td>
</tr>
</tbody>
</table>

Vegetables

<table>
<thead>
<tr>
<th>Almonds</th>
<th>Mushrooms</th>
<th>Tomatoes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avocado</td>
<td>Broccoli</td>
<td>Carrots</td>
</tr>
</tbody>
</table>

Minerals

<table>
<thead>
<tr>
<th>Rock</th>
<th>Gemstone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Granite</td>
<td>Amethyst</td>
</tr>
<tr>
<td>Quartz</td>
<td>Topaz</td>
</tr>
</tbody>
</table>

Chemical Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Symbol</th>
<th>Atomic Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iron</td>
<td>Fe</td>
<td>26</td>
</tr>
<tr>
<td>Carbon</td>
<td>C</td>
<td>6</td>
</tr>
</tbody>
</table>

SI Units

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meter</td>
<td>m</td>
</tr>
<tr>
<td>Second</td>
<td>s</td>
</tr>
<tr>
<td>Kelvin</td>
<td>K</td>
</tr>
</tbody>
</table>

Information Symbols

- 8 Examples: sample-multi2.pdf (pages 5 to 8)
- 2 Examples: sample-multi2.pdf (pages 5 to 8)
Package Option Summary

Most options are in the form \( \langle \text{option} \rangle = \langle \text{value} \rangle \) and may have a default if \( \langle \text{value} \rangle \) 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.

acssupp

Load the glossaries-acssupp 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.

**C**

counter=⟨value⟩

Sets the default location counter to \( \langle \text{value} \rangle \) (which must be the name of a counter). May be overridden on an individual basis using the counter option in commands like \texttt{\textbackslash gls} and \texttt{\textbackslash glsadd}. Provided by glossaries.

counterwithin=⟨counter name⟩

Automatically sets the option entrycounter =\{true\} and indicates the master counter for glossaryentry. Provided by glossaries.

**D**

debug=⟨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.
docdef\langle value\rangle
Determines whether entries can be defined in the document environment; the \langle value\rangle may be one of: \texttt{false} (entries must be defined in the preamble), \texttt{true} (entries may be defined in the document environment), \texttt{restricted} (entries may only be defined in the document environment if the definition comes before all glossaries and before any reference to the entry).
\texttt{Provided by glossaries-extra. Not relevant with bib2gls.}

entrycounter\langle boolean\rangle
If true, creates the glossaryentry counter and each main (level 0) glossary entry will be numbered (which can be referenced with \texttt{\glsrefentry} or \texttt{\glsxtrpageref}).
\texttt{Provided by glossaries.}

esclocations\langle boolean\rangle
If true, glossaries tries to escape special characters from the locations.
\texttt{Provided by glossaries. Not relevant with bib2gls.}

hyperfirst\langle boolean\rangle
If false, terms on first use don’t have hyperlinks unless explicitly set (with glossaries-extra, the \texttt{nohyperfirst} category attribute can selectively apply this).
\texttt{Provided by glossaries.}

index
Defines the index glossary and \texttt{\newterm}.\texttt{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.
\texttt{Provided by glossaries-extra.}

indexcrossrefs\langle boolean\rangle
If true, at the end of the document automatically index cross-referenced entries that haven’t been marked as used.
\texttt{Provided by glossaries-extra. Not relevant with bib2gls.}

indexonlyfirst\langle boolean\rangle
If true, only performs indexing on first use.
\texttt{Provided by glossaries.}

makeindex
Write the indexing information using \texttt{makeindex}'s format.
\texttt{Provided by glossaries. Not relevant with bib2gls.}

nogroupskip\langle boolean\rangle
If true, suppresses the visual separation between letter groups in glossary styles that support this option.
\texttt{Provided by glossaries.}

nohypertypes\langle list\rangle
Suppress hyperlinks for the listed glossary types.
\texttt{Provided by glossaries.}

nolangwarn
Suppresses warnings generated by missing language modules.
\texttt{Provided by glossaries.}

nolist
Prevents the glossary-list package (which provides the list styles) from being automatically loaded.
\texttt{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, suppress 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, \printunrtglossary and \printnoidxglossary is boolean); with bib2gls you can use save-locations={false} instead.
Provided by glossaries-extra.

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), nolabel (numbered but no label), autolabel (numbered with automatic label), nameref (unnumbered but labelled). If no value is given nolabel 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.
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 indexed 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.
shortcuts={value}

Sets up shortcut 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}).

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.

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).

style={name}

Sets the default glossary style to {name}.

toc={boolean}

If true (default for glossaries-extra), automatically add each glossary to the table of contents.

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).

ucmark={boolean}

If true, converts the glossary mark (used in page headings) to upper case with \MakeTextUppercase.

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.

\[ xindy=\langle\text{settings}\rangle \]

Write the indexing information using xindy’s format where the optional \langle\text{settings}\rangle 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={}.<\settings>

Provided by glossaries. Not relevant with bib2gls.

\[ xindynoglsnumbers \]

Equivalent to xindy={glsnumbers=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.

\}
  Produces the close brace symbol \}.

\{
  Produces the open brace symbol \{.

\"{⟨character⟩}
  Puts an umlaut accent over ⟨character⟩.

\#
  Produces the hash symbol #.

\%
  Produces the percent symbol %.

\&
  Produces the ampersand symbol &.

\'{⟨character⟩}
  Puts an acute accent over ⟨character⟩.

\,
  Thin space.

\.⟨character⟩
  Puts a dot accent over ⟨character⟩.

@
  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.
General Command Summary

\texttt{\firstofone{code}} \quad \text{kernel command*}
Does (code).

\texttt{\for(cs):=\langle list\rangle\do{\langle code\rangle}} \quad \text{kernel command*}
Iterates over each item in the comma-separated (list), and on each iteration sets (cs) to the current element and performs (code).

\texttt{\gls@hypergroup\{type\}\{group id\}} \quad \text{glossary-hypernav}
Command written to the .aux file that identifies that the given group was used in the glossary on the previous run.

\texttt{\gobble{\langle code\rangle}} \quad \text{kernel command*}
Does nothing (the argument is discarded).

\texttt{\istfilename{\langle filename\rangle}} \quad \text{glossaries}
Identifies the style file in the .aux file for the benefit of external tools like makeglossaries and makeglossaries-lite.

\texttt{\newline{\langle len\rangle}} \quad \text{kernel command*}
Starts a new row in a tabular or array context with an extra vertical space of length (len) above it (starred form prohibits a page break).

\texttt{\^\{\langle character\rangle\}} \quad \text{kernel command*}
Puts a circumflex accent over (character).

\texttt{\_} \quad \text{kernel command*}
Produces the underscore symbol _.

\texttt{\␣} \quad \text{kernel command*}
Produces an inter-word space.

A

\texttt{\AA} \quad \text{kernel command*}
Produces the upper case A-ring character Å.

\texttt{\aa} \quad \text{kernel command*}
Produces the lower case a-ring character å.

\texttt{\ab}\{\langle options\rangle\}\{\langle label\rangle\}\{\langle insert\rangle\} \quad \text{glossaries-extra shortcuts=abbreviations}
Equivalent to \gls.

\texttt{\abbreviationname} \quad \text{glossaries-extra}
Language-sensitive name used for the title of the glossary created with the abbreviations package option.

\texttt{\abbrvpluralsuffix} \quad \text{glossaries-extra*}
The style sensitive suffix used to construct the default plural for the short form of abbreviations.

\texttt{\ac}\{\langle options\rangle\}\{\langle label\rangle\}\{\langle insert\rangle\} \quad \text{glossaries-extra shortcuts}
Equivalent to \gls.
\acro nameof glossaries
Language-sensitive name used for the title of the glossary created with the \acr or \acronym package option.

\acro typeof 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 kernel command*
Produces the upper case \textit{Æ}-ligature.

\ae kernel command*
Produces the lower case \textit{æ}-ligature.

\alph{\langle counter \rangle} kernel command
Displays the given counter as an alphabetic character from “a” to “z”.

\Alpha glossaries-extra-bib2gls*
Greek letter alpha A.

\alpha kernel command* (maths mode)
Greek letter alpha \textit{α}.

\alsoname language packages
Language sensitive “see also” text.

\approx kernel command* (maths mode)
Approximate symbol \textit{≈}.

\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} kernel command
Perform \langle code \rangle at the end of the document.

\autoref{\langle id \rangle} hyperref
Cross-reference with textual tag inferred from the associated counter.

\backmatter book-like classes
Switches to back matter.

\bfseries kernel command
Switch to bold (until end of current scope).
General Command Summary

\bibglsaliassep
Separator between alias cross-reference and location list.

\bibglsampersandchar
Expands to a literal ampersand 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 BIBTEX’s contributor syntax.

\bibglscontributorlist{⟨list⟩}{⟨number⟩}
Used to markup a list of names from a field that was converted from BIBTEX’s contributor syntax.

\bibglsdate{⟨year⟩}{⟨month⟩}{⟨day-of-month⟩}{⟨day-of-week⟩}{⟨day-of-year⟩}{⟨era⟩}
{⟨original⟩}
Used to markup a date converted from a field value.

\bibglsdatetime{⟨year⟩}{⟨month⟩}{⟨day-of-month⟩}{⟨day-of-week⟩}{⟨day-of-year⟩}{⟨era⟩}
{⟨hour⟩}{⟨minute⟩}{⟨second⟩}{⟨millisec⟩}{⟨dst⟩}{⟨zone⟩}{⟨original⟩}
Used to markup a date-time instance converted from a field value.

\bibglsdatetimegroup{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}{⟨hh⟩}{⟨mm⟩}{⟨ss⟩}{⟨zone⟩}{⟨title⟩}{⟨group-id⟩}{⟨type⟩}
Expands to the date-time group label.

\bibglsdatetimegrouptitle{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}{⟨hh⟩}{⟨mm⟩}{⟨ss⟩}{⟨zone⟩}{⟨title⟩}{⟨group-id⟩}{⟨type⟩}
Expands to the date-time group title.

\bibglsdatetimeremainder{⟨millisec⟩}{⟨dst⟩}{⟨zone⟩}{⟨original⟩}
Used internally to pick up the final four arguments of \bibglsdatetime.

\bibglsdelimN
Separator for individual locations (except last).

\bibglsdollarchar
Expands to a literal dollar character.

\bibglsdualprefixlabel{⟨prefix⟩}
Hook provided to pick up the dual prefix, if required.
General Command Summary

\bibglsemptygroup{(type)}
Expands to the empty group label.

\bibglsemptygrouptitle{(type)}
Expands to the 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.

\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.

\bibglslastDelimN
Separator before the last location (where there is more than one location).

\bibglslettergroup{(title)}{(letter)}{(id)}{(type)}
Expands to the letter group label.

\bibglslettergrouptitle{(title)}{(letter)}{(id)}{(type)}
Expands to the letter group title.

\bibglslocationgroup{(n)}{(counter)}{(list)}
Location group encapsulator.
General Command Summary

\bibglslocationgroupsep
Location group separator.

\bibglslocprefix\{n\}
Location list prefix.

\bibglslocsuffix\{n\}
Location list suffix.

\bibglslowercase\{text\}
Converts \text{text} to lower case.

\bibglsnewabbreviation\{label\}\{options\}\{short\}\{long\}
Defines terms provided with \texttt{@abbreviation}.

\bibglsnewacronym\{label\}\{options\}\{short\}\{long\}
Defines terms provided with \texttt{@acronym}.

\bibglsnewbibtexentry\{label\}\{options\}\{name\}\{description\}
Defines terms provided with \texttt{@bibtexentry}.

\bibglsnewcontributor\{label\}\{options\}\{name\}\{description\}
Defines terms provided with \texttt{@contributor}.

\bibglsnewdualabbreviation\{label\}\{options\}\{short\}\{long\}
Defines terms provided with \texttt{@dualabbreviation}.

\bibglsnewdualabbreviationentry\{label\}\{options\}\{short\}\{long\}
\{description\}
Defines primary terms provided with \texttt{@dualabbreviationentry}.

\bibglsnewdualabbreviationentrysecondary\{label\}\{options\}\{short\}\{long\}
\{description\}
Defines secondary terms provided with \texttt{@dualabbreviationentry}.

\bibglsnewdualacronym\{label\}\{options\}\{short\}\{long\}
Defines terms provided with \texttt{@dualacronym}.

\bibglsnewdualentry\{label\}\{options\}\{name\}\{description\}
Defines terms provided with \texttt{@dualentry}.

\bibglsnewdualentryabbreviation\{label\}\{options\}\{short\}\{long\}
\{description\}
Defines primary terms provided with (deprecated) \texttt{@dualentryabbreviation}.

\bibglsnewdualentryabbreviationsecondary\{label\}\{options\}\{short\}\{long\}
\{description\}
Defines secondary terms provided with (deprecated) \texttt{@dualentryabbreviation}.

\bibglsnewdualindexabbreviation\{label\}\{\texttt{dual-label}\}\{options\}\{name\}\{short\}\{long\}\{description\}
Defines primary terms provided with \texttt{@dualindexabbreviation}.
Defines secondary terms provided with \texttt{\textbackslash @spawndualindexabbreviation}
\texttt{\textbackslash dualindexabbreviationsecondary}

\begin{verbatim}
\bibglsnewdualindexabbreviationsecondary\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle short\rangle}\{\langle long\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines secondary terms provided with \texttt{\textbackslash dualindexabbreviation}.

Defines primary terms provided with \texttt{\textbackslash dualindexentry}.

\begin{verbatim}
\bibglsnewdualindexentry\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines primary terms provided with \texttt{\textbackslash dualindexentry}.

\begin{verbatim}
\bibglsnewdualindexentrysecondary\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines secondary terms provided with \texttt{\textbackslash dualindexentry}.

\begin{verbatim}
\bibglsnewdualindexnumber\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle symbol\rangle}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines primary terms provided with \texttt{\textbackslash dualindexnumber}.

\begin{verbatim}
\bibglsnewdualindexnumbersecondary\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines secondary terms provided with \texttt{\textbackslash dualindexnumber}.

\begin{verbatim}
\bibglsnewdualindexsymbol\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle symbol\rangle}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines primary terms provided with \texttt{\textbackslash dualindexsymbol}.

\begin{verbatim}
\bibglsnewdualindexsymbolsecondary\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines secondary terms provided with \texttt{\textbackslash dualindexsymbol}.

\begin{verbatim}
\bibglsnewdualnumber\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash dualnumber}.

\begin{verbatim}
\bibglsnewdualsymbol\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash dualsymbol}.

\begin{verbatim}
\bibglsnewentry\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash entry}.

\begin{verbatim}
\bibglsnewindex\{\langle label\rangle\}{\langle options\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash index}.

\begin{verbatim}
\bibglsnewindexplural\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash index}.

\begin{verbatim}
\bibglsnewnumber\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash number}.

\begin{verbatim}
\bibglsnewprogenitor\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash progenitor}.

\begin{verbatim}
\bibglsnewspawnabbreviation\{\langle label\rangle\}{\langle options\rangle}\{\langle short\rangle\}{\langle long\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash spawnabbreviation}.

\begin{verbatim}
\bibglsnewspawnacronym\{\langle label\rangle\}{\langle options\rangle}\{\langle short\rangle\}{\langle long\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash spawnacronym}.

\begin{verbatim}
\bibglsnewspawnindex\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines terms provided with \texttt{\textbackslash spawnindex}.

\begin{verbatim}
\bibglsnewspawnindexsecondary\{\langle label\rangle\}{\langle options\rangle}\{\langle name\rangle\}{\langle description\rangle}
\end{verbatim}

\texttt{\textbackslash bib2gls}

Defines secondary terms provided with \texttt{\textbackslash spawnindex}.
General Command Summary

\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 @progenitor or @spawnindex.

\bibglsnewspawnedindexplural\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}
\bib2gls
Defines terms spawned from @spawnindexplural.

\bibglsnewspawnednumber\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms spawned from @spawnnumber.

\bibglsnewspawnedsymbol\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms spawned from @spawnsymbol.

\bibglsnewspawnentry\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms provided with @spawnentry.

\bibglsnewspawnindex\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms provided with @spawnindex.

\bibglsnewspawnindexplural\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms provided with @spawnindexplural.

\bibglsnewspawnnumber\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms provided with @spawnnumber.

\bibglsnewspawnedsymbol\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms provided with @spawnsymbol.

\bibglsnewsymbol\{\langle label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle description \rangle\}
\bib2gls
Defines terms provided with @symbol.

\bibglsnewtertiaryindexabbreviationentry\{\langle label \rangle\}\{\langle dual-label \rangle\}\{\langle options \rangle\}\{\langle name \rangle\}\{\langle short \rangle\}\{\langle long \rangle\}\{\langle description \rangle\}
\bib2gls
Defines primary terms provided with @tertiaryindexabbreviationentry.

\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\}
\bib2gls
Defines secondary and tertiary terms provided with @tertiaryindexabbreviationentry.

\bibglsnumbergroup\{\langle value \rangle\}\{\langle id \rangle\}\{\langle type \rangle\}
\bib2gls
Expands to the number group label.

\bibglsnumbergrouptitle\{\langle value \rangle\}\{\langle id \rangle\}\{\langle type \rangle\}
\bib2gls
Expands to the number group title.
General Command Summary

\bibglsothergroup\{\langle character\rangle\}\{id\}\{type\}\}
  Expands to the non-letter group label.
\bibglsothergrouptitle\{\langle character\rangle\}\{id\}\{type\}\}
  Expands to the non-letter group title.
\bibglspagename
  Name used for single page.
\bibglspagesname
  Name used for multiple pages.
\bibglspassim
  Passim range suffix.
\bibglspassimname
  Name used by passim range suffix.
\bibglspostlocprefix
  Location list post prefix.
\bibglprimary\{n\}\{locations\}
  Location list encapsulator used in the primarylocations field.
\bibglprimaryprefixlabel\{prefix\}
  Hook provided to pick up the primary prefix, if required.
\bibglrange\{start\}\delimR\{end\}
  Explicit range format.
\bibglsseealsosep
  Separator between seealso cross-references and location list.
\bibglsseesep
  Separator between see cross-references and location list.
\bibglsetdategrouptitle\{(YYYY)\}(MM)\{DD\}(G)\{title\}\{group-id\}\
  \{type\}\}
  Sets the date (no time) group title.
\bibglsetdatetimegrouptitle\{(YYYY)\}(MM)\{DD\}\{hh\}\{mm\}\{ss\}\{zone\}\
  \{title\}\{group-id\}\{type\}\}
  Sets the date-time group title.
\bibglsetemptygrouptitle\{type\}\}
  Sets the empty group title.
\bibglsetlastgrouptitle\{cs\}\{specs\}\}
  Sets the last group title.
\bibglsetlettergrouptitle\{\langle title\rangle\}\{\langle letter\rangle\}\{id\}\{type\}\}
  Sets the letter group title.
\bibglsetnumbergrouptitle\{\langle value\rangle\}\{id\}\{type\}\}
  Sets the number group title.
\bibglssetothergrouptitle\{\langle character\rangle\}{\langle id\rangle}\{\langle type\rangle\}
Sets the non-letter group title.

\bibglssettimegrouptitle\{\langle hh\rangle\}{\langle mm\rangle}\{\langle ss\rangle\}{\langle zone\rangle}\{\langle title\rangle\}{\langle group-id\rangle}\{\langle type\rangle\}
Sets the time (no date) group title.

\bibglssetunicodedegrouptitle\{\langle label\rangle\}{\langle character\rangle}\{\langle id\rangle\}\{\langle type\rangle\}
Sets the Unicode script, category or character code title.

\bibglssetwidest\{\langle level\rangle\}{\langle name\rangle}
Sets the widest name.

\bibglssetwidestfallback\{\langle glossary list\rangle\}
Fallback used instead of \bibglssetwidest in the event that bib2gls can’t determine the widest name, where \langle glossary list\rangle is a comma-separated list of glossary labels.

\bibglssetwidestfortype\{\langle type\rangle\}{\langle level\rangle}\{\langle name\rangle\}
Sets the widest name for the given glossary type.

\bibglssetwidestfortypefallback\{\langle type\rangle\}
Fallback used instead of \bibglssetwidestfortype in the event that bib2gls can’t determine the widest name.

\bibglssetwidesttoplevelfallback\{\langle glossary list\rangle\}
Fallback used instead of \bibglssetwidest 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.

\bibglssetwidesttoplevelfortypefallback\{\langle type\rangle\}
Fallback used instead of \bibglssetwidestfortype in the event that bib2gls can’t determine the widest name where there are only top-level entries.

\bibglssupplemental\{\langle n\rangle\}{\langle list\rangle}
Supplemental list encapsulator.

\bibglssupplementalsep
Separator between main and supplementary locations.

\bibglssupplementalsublist\{\langle n\rangle\}{\langle external document\rangle}\{\langle list\rangle\}
Supplemental sub-list encapsulator.

\bibglssupplementalsubsep
Separator between supplementary sub-lists.

\bibglstertiaryprefixlabel\{\langle prefix\rangle\}
Hook provided to pick up the tertiary prefix, if required.

\bibglstime\{\langle hour\rangle\}{\langle minute\rangle}\{\langle second\rangle\}{\langle milliseck\rangle}\{\langle zone\rangle\}{\langle original\rangle\}
Used to markup a time converted from a field value.

\bibglstimegroup\{\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 label.
\bibglstimegrouptitle{(hh)}{(mm)}{(ss)}{(zone)}{(group-id)}{(type)}
bib2gls
Expands to the time group title.

\bibglistitlecase{(text)}
bib2gls*
Converts \textit{(text)} to title case.

\bibglsunderscorechar
bib2gls*
Expands to a literal underscore character.

\bibglsunicodegroup{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}
bib2gls
Expands to the Unicode script or category label or character code.

\bibglsunicodetitle{⟨label⟩}{⟨character⟩}{⟨id⟩}{⟨type⟩}
bib2gls
Expands to the Unicode script or category label or character code.

\bibglsuppercase{(text)}
bib2gls*
Converts \textit{(text)} to upper case.

\bibglsuseabbrvfont{(text)}{(category)}
bib2gls
Ensures that the given text is formatted according to the given category’s short format.

\bibglsusealias{⟨label⟩}
bib2gls
Display the \textit{alias} cross-reference for given entry.

\bibglsuselongfont{(text)}{(category)}
bib2gls
Ensures that the given text is formatted according to the given category’s long format.

\bibglsusesee{⟨label⟩}
bib2gls
Display \textit{see} cross-reference list for given entry.

\bibglsuseseealso{⟨label⟩}
bib2gls
Display the \textit{seealso} cross-reference list for given entry.

\bibliography{(file list)}
kernell command*
Display bibliography created by \TeX{.}

\bigoperatornamefmt{(text)}
Example command.

\boldsymbol{(symbol)}
amsmath
Renders given maths symbol in bold if supported by the current font.

\bottomrule
booktabs
Horizontal rule for the bottom of a tabular-like environment.

C

\c{(character)}
kernell command*
Puts a cedilla accent over \textit{(character)}.

\capitalisewords{(text)}
mfirstuc v1.06+
Converts the first letter of each word to upper case using \texttt{\textbackslash makefirstuc}. 515
\caption{(list title)\{title\}}  \hspace*{1cm} kernel command
Caption title.
\ce{\textit{(formula)}}  \hspace*{1cm} mhchem
Displays the chemical formula.
\chapter{(toc title)\{title\}}  \hspace*{1cm} \textit{book or report classes}
Chapter heading.
\chapter*{\{title\}}  \hspace*{1cm} \textit{book or report classes}
Unnumbered chapter heading.
\char{(number)}  \hspace*{1cm} \TeX\ primitive
Accesses the character identified by \textit{(number)} (use \texttt{\char"\{hex\}} if the number is hexadecimal).
\citation{\{label\}}  \hspace*{1cm} kernel command
Written to the .aux file on each occurrence of \texttt{\cite}.
\cite{\{label\}}  \hspace*{1cm} kernel command
Cross-reference a bibliographic citation.
\cjkname{\{CJK characters\}}  \hspace*{1cm} 
Example command.
\color{\{model\}\{spec\}}  \hspace*{1cm} color
Switches the current font colour.
\csuse{\{cs-name\}}  \hspace*{1cm} etoolbox
Uses the control sequence whose name is given by \textit{(cs-name)} or does nothing if the command isn’t defined.
\currentglossary  \hspace*{1cm} \textit{glossaries}
Defined within the glossary to the current glossary type, this has no meaning outside of the glossary list.
\DeclareOptions{\{name\}\{code\}}  \hspace*{1cm} kernel command
Declares an option with the given \textit{(name)}.
\DeclareOptions*{\{code\}}  \hspace*{1cm} kernel command
Indicates what to do with unknown options.
\def{\{cs\}\{syntax\}\{definition\}}  \hspace*{1cm} \TeX\ primitive
Defines the control sequence \textit{(cs)}, without checking if the command already exists.
\delimN  \hspace*{1cm} \textit{glossaries}
Used to delimited individual locations.
\delimR  \hspace*{1cm} \textit{glossaries}
Used as a separator between the start and end locations of a range.

516
General Command Summary

\descriptionname  
Language-sensitive name used for the description header for some glossary styles.

\detokenize{\{general text\}}  
$\varepsilon$-\LaTeX{} 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.

\dGlspl{\{options\}}{\{label\}}{\{insert\}}  
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\}}  
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  
kernel command*
Produces the upper case eth Đ.

\dh  
kernel command*
Produces the lower case eth đ.

\diamondsuit  
kernel command* (maths mode)
Diamond symbol ($\diamondsuit$).

\displaystyle  
kernel command (maths mode)
Switch to display maths style.

\DJ  
kernel command*
Produces the upper case d-stroke Đ.

\dj  
kernel command*
Produces the lower case d-stroke đ.

\DTLandname  
datatool-base* v2.28+
Used in the definition of \DTLlistformatlastsep.

\DTLformatlist{\{list\}}  
datatool-base* v2.28+
Formats a comma-separated list.
General Command Summary

\DTListformatlastsep datatool-base* v2.28+
Used by \DTListformatlist to separate the last two items in the list.

\DTListformatoxford datatool-base* v2.28+
Insert before \DTListformatlastsep if the list has three or more items.

\DTMDisplaydate{⟨year⟩}{⟨month⟩}{⟨day⟩}{⟨dow⟩} datetime2
Formats the given date where all arguments are numeric.

E

\edef{⟨cs⟩}{⟨syntax⟩}{⟨definition⟩} \TeX primitive*
Defines the control sequence ⟨cs⟩ to the full expansion of ⟨definition⟩, without checking if
the command already exists.

\eglsupdatewidest{⟨level⟩}{⟨text⟩} glossaries-extra-stylemods v1.23+
As \glsupdatewidest but expands ⟨text⟩.

\em kernel command
Switch to emphasized font (until end of current scope).

\emph{⟨text⟩} kernel command
Emphasizes the given text (italic or slanted if the surrounding font is upright, otherwise
upright font is used).

\endfoot longtable
Ends the footer section.

\endhead longtable
Ends the header section.

\ensuremath{⟨maths⟩} 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.

F

\footnote{⟨number⟩}{⟨text⟩} kernel command
Displays the given text as a footnote.

\forall kernel command* (maths mode)
For all symbol (∀).

\forallglossaries{⟨glossary-list⟩}{⟨cs⟩}{⟨body⟩} 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.
General Command Summary

\forallglsentries[\langle glossary-list \rangle](cs)\{body\} \quad \text{glossaries}
Iterates over all entries defined in the comma-separated \langle glossary-list \rangle (or all defined non-ignored glossaries if the optional argument is omitted) and perform \langle body \rangle where you can use the control sequence \langle cs \rangle to reference the current entry label.

\forglstentries[\langle type \rangle](cs)\{body\} \quad \text{glossaries}
Iterates over all entries defined in the glossary identified by \langle type \rangle (or the default, if \langle type \rangle is omitted) and perform \langle body \rangle where you can use the control sequence \langle cs \rangle to reference the current entry label.

\forlistloop[\langle handler \rangle](cs)\{list \} \quad \text{etoolbox}
Iterates over the internal list given by the command \langle list \rangle and performs \langle handler \rangle \{ \langle element \rangle \} for each element.

\frontmatter \quad \text{book-like classes}
Switches to front matter.

G

\glolinkprefix \quad \text{glossaries}
Target name prefix used in entry hyperlinks.

\glossariesextrasetup\{\langle key=value \ list \rangle\} \quad \text{glossaries-extra}
Applies the extension \text{glossaries-extra} options that are allowed to be changed after the package has loaded.

\glossaryheader \quad \text{glossaries}
Implemented at the start of a glossary (modified by \text{glossary} styles).

\glossaryname \quad \text{glossaries or language packages}
Language-sensitive name used for the title of the default main glossary.

\glossarypostamble \quad \text{glossaries}
The postamble that’s placed after each glossary.

\glossarypreamble \quad \text{glossaries}
The preamble for all glossaries except those that have the preamble explicitly set with \text{apptoglossarypreamble}.

\glossentry\{\langle label \rangle\}\{\langle location \ list \rangle\} \quad \text{glossaries v3.08a+}
Used in the glossary to display a top-level entry.

\Glossentrydesc\{\langle label \rangle\} \quad \text{glossaries}
Like \text{\glossentrydesc} but converts the first letter to upper case.

\glossentrydesc\{\langle label \rangle\} \quad \text{glossaries}
Used by \text{glossary} styles to display the description.

\Glossentryname\{\langle label \rangle\} \quad \text{glossaries}
Like \text{\glossentryname} but converts the first letter to upper case.
General Command Summary

\glossentryname{⟨label⟩}
glossaries
Used by glossary styles to display the name.

\glossentrynameother{⟨label⟩}{⟨field⟩}
glossaries-extra v1.22+
Acts like \glossentryname (obeys glossname and glossnamefont or \glsnamefont and the post-name hook) but uses the given ⟨field⟩ instead of the name field.

\Glossentrysymbol{⟨label⟩}
glossaries
Like \glossentryname but converts the first letter to upper case.

\glossxtrsetpopts
glossaries-extra
Glossary hook that uses \glsxtrsetpopts to enable hyperlinks by default for \glsxtrp.

\GLS{⟨options⟩}{⟨label⟩}{⟨insert⟩}
glossaries
As \gls but converts the link text to upper case.

\Gls{⟨options⟩}{⟨label⟩}{⟨insert⟩}
glossaries
As \gls but converts the first letter of the link text to upper case.

\gls{⟨options⟩}{⟨label⟩}{⟨insert⟩}
glossaries
On first use displays the first use text (the value of the first field for general entries) and on subsequent use displays the subsequent use text (the value of the 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 \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 \emph.

\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 \glsabbrvdefaultfont.

\glsabbrvscfont{⟨text⟩}
glossaries-extra v1.17+
Used with “sc” abbreviation styles to format the short form using \textsc.

\glsabbrvsmfont{⟨text⟩}
glossaries-extra v1.17+
Used with “sm” abbreviation styles to format the short form using \textsmaller.

\glsabbrvuserfont{⟨text⟩}
glossaries-extra v1.04+
Used with “user” abbreviation styles to format the short form. The default definition just uses \glsabbrvdefaultfont.
\Glsaccessdesc\{\langle label \rangle\} \quad \text{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 \Glsentrydesc).

\glsaccessdesc\{\langle label \rangle\} \quad \text{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 \glssentrydesc).

\Glsaccessdescplural\{\langle label \rangle\} \quad \text{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 \Glsentrydescplural).

\glsaccessdescplural\{\langle label \rangle\} \quad \text{glossaries-extra*}

Expands to the value of the \texttt{descriptionplural} field with the accessibility support for that field, if provided (otherwise behaves the same as \glssentrydescplural).

\glssdisplay\{\langle field \rangle\}\{\langle text \rangle\}\{\langle label \rangle\} \quad \text{glossaries-access}

Displays \langle text \rangle with the accessibility support provided by \Glsentry\{\langle field \rangle\}access\{\langle label \rangle\}.

\Glsaccessfirst\{\langle label \rangle\} \quad \text{glossaries-extra*}

Expands to the value of the \texttt{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\{\langle label \rangle\} \quad \text{glossaries-extra*}

Expands to the value of the \texttt{first} field with the accessibility support for that field, if provided (otherwise behaves the same as \glssentryfirst).

\Glsaccessfirstplural\{\langle label \rangle\} \quad \text{glossaries-extra*}

Expands to the value of the \texttt{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\{\langle label \rangle\} \quad \text{glossaries-extra*}

Expands to the value of the \texttt{firstplural} field with the accessibility support for that field, if provided (otherwise behaves the same as \glssentryfirstplural).

\Glsaccesslong\{\langle label \rangle\} \quad \text{glossaries-extra*}

Expands to the value of the \texttt{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\{\langle label \rangle\} \quad \text{glossaries-extra*}

Expands to the value of the \texttt{long} field with the accessibility support for that field, if provided (otherwise behaves the same as \glssentrylong).
General Command Summary

\Glsaccesslongpl\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{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\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{longplural} field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentrylongpl).

\Glsaccessname\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{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\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{name} field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryname).

\Glsaccessplural\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{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\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{plural} field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryplural).

\Glsaccessshort\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{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\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{short} field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryshort).

\Glsaccessshortpl\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{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\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{shortplural} field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentryshortpl).

\Glsaccesssymbol\{\textit{\texttt{label}}\} \textit{glossaries-extra}*
Expands to the value of the \texttt{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 \Glsentriesymbol).
\glsaccesssymbol{⟨label⟩}
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⟩}
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⟩}
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).

\Glsaccesstext{⟨label⟩}
glossaries-extra*
Expands to the value of the 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{⟨label⟩}
glossaries-extra*
Expands to the value of the text field with the accessibility support for that field, if provided (otherwise behaves the same as \glsentrytext).

\glsacctext{⟨accessible text⟩}{⟨text⟩}
glossaries-accsupp
Used by the accessibility support to interface with the accsupp package (use \xglsaccsupp if ⟨text⟩ needs to be fully expanded first).

\glsadd{⟨options⟩}{⟨label⟩}
glossaries
Indexes the entry without displaying any text.

Options:

\texttt{counter}=⟨⟨counter-name⟩⟩
glossaries
Sets the counter to use for the record

\texttt{format}=⟨⟨encap⟩⟩
glossaries
Sets the ENCAP for the record to ⟨encap⟩, optionally with the start or end range markers

\texttt{theHvalue}=⟨⟨value⟩⟩
glossaries-extra v1.14+
The hyperlink target corresponding to the value of thevalue, if appropriate

\texttt{thevalue}=⟨⟨value⟩⟩
glossaries-extra v1.14+
Overrides the record value so that it’s the given ⟨value⟩ not obtained from the associated counter

\glsaddall{⟨options⟩}
glossaries
Iterates over all entries defined for all glossaries (or for the sub-list provided by types=⟨⟨list⟩⟩ in the options) and performs \glsadd{⟨options⟩} for each entry. This command isn’t suitable for use with bib2gls. Use the selection option instead.

\glsaddallunused{⟨list⟩}
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 format set
to glsignore. This command isn’t suitable for use with bib2gls. Use the selection option instead.

\glsaddeach[⟨options⟩]{⟨label list⟩}

Indexes each entry identified in the comma-separated list of labels without displaying any text.

\glsaddkey{⟨key⟩}{⟨default value⟩}{⟨no link cs⟩}{⟨link ucfirst cs⟩}

Adds a new key for use in \newglossaryentry and associated commands to access it.

\glsaddstoragekey{⟨key⟩}{⟨default value⟩}{⟨no link cs⟩}

Adds a new key for internal use that can be set in \newglossaryentry.

\glsautoprefix

Prefix used for the automatically labelling triggered by the numberedsection={autolabel} option.

\glspars

Expands to a literal backslash \ character.

\glscapturedgroup

Expands to \string$.

\glscategory{⟨label⟩}

Expands to the value of the category field for the entry identified by ⟨label⟩ or nothing if the entry hasn’t been defined.

\glsclosebrace

Expands to a literal close brace } character.

\glscurrententrylabel

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

Only for use in the ⟨true⟩ part of \ifglishasfield or \glsxtrifhasfield, this expands to the field value.

\glsdefaulttype

The default glossary type.

\glsdefpostdesc{⟨category⟩}{⟨definition⟩}

Define the post-description hook \glsxtrpostdesc{category} for the given category.

\glsdefpostlink{⟨category⟩}{⟨definition⟩}

Define the post-link hook \glsxtrpostlink{category} for the given category.

\glsdefpostname{⟨category⟩}{⟨definition⟩}

Define the post-name hook \glsxtrpostname{category} for the given category.

\glsdesc[⟨options⟩]{⟨label⟩}{⟨insert⟩}

Links to the entry’s definition in the glossary with the link text obtained from the description field without altering the first use flag.
\glsdescriptionaccessdisplay{\text}{\label} \glossaries-access
Displays \text with the accessibility support provided by \glsentrydescaccess{\label}.

\glsdescriptionpluralaccessdisplay{\text}{\label} \glossaries-access
Displays \text with the accessibility support provided by \glsentrydescpluralaccess{\label}.

\glsdescwidth \glossary-long and \glossary-super
Length register used by the tabular styles to specify the width of the description column.

\glsdisablehyper \glossaries
Disables the creation of hyperlinks and targets for the glossary commands that support them (automatically implemented if \hyperref isn’t loaded before \glossaries).

\glsdisp{\options}{\label}{\text} \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.

\glsdoifexists{\label}{\code} \glossaries
If the entry given by \label exists, \code is done, otherwise an error (or warning with \glossaries-extra’s undeclaration={warn} option) is triggered.

\glsdoifexistsordonoexists{\label}{\code}{\else code} \glossaries
If the entry given by \label exists, \code is done, otherwise an error (or warning with \glossaries-extra’s undeclaration={warn} option) is triggered and \else code is done.

\glsdoifnoexists{\label}{\code} \glossaries
If the entry given by \label doesn’t exist, \code is done, otherwise an error (or warning with \glossaries-extra’s undeclaration={warn} option) is triggered.

\glsdoifnoexistsordonoexists{\label}{\code}{\else code} \glossaries
If the entry given by \label doesn’t exist, \code is done, otherwise an error (or warning with \glossaries-extra’s undeclaration={warn} option) is triggered and \else code is done.

\glsenablehyper \glossaries
Enables the creation of hyperlinks and targets for the glossary commands that support them (automatically implemented if \hyperref is loaded before \glossaries).

\glsentryaccess{\label} \glossaries-access
Expands to the value of the \access field.

\glsentrycounterlabel \glossaries
Governs the way the \glossaryentry counter is displayed by \glsentryitem.

\GlsEntryCounterLabelPrefix \glossaries v4.38+
Used as a prefix in the \label command automatically implemented by the \entrycounter and \subentrycounter options.

\Glsentrydesc{\label} \glossaries*
Displays the value of the \description field with the first letter converted to upper case.
General Command Summary

```latex
\glsentrydesc{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Expands to the value of the description field.}
```

```latex
\glsentrydescaccess{⟨label⟩} \quad \text{glossaries-access}
\quad \text{Expands to the value of the descriptionaccess field.}
```

```latex
\Glsentrydescplural{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Displays the value of the descriptionplural field with the first letter converted to upper case.}
```

```latex
\glsentrydescplural{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Expands to the value of the descriptionplural field.}
```

```latex
\glsentrydescpluralaccess{⟨label⟩} \quad \text{glossaries-access}
\quad \text{Expands to the value of the descriptionpluralaccess field.}
```

```latex
\Glsentryfirst{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Displays the value of the first field with the first letter converted to upper case.}
```

```latex
\glsentryfirst{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Expands to the value of the first field.}
```

```latex
\glsentryfirstaccess{⟨label⟩} \quad \text{glossaries-access}
\quad \text{Expands to the value of the firstaccess field.}
```

```latex
\Glsentryfirstplural{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Displays the value of the firstplural field with the first letter converted to upper case.}
```

```latex
\glsentryfirstplural{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Expands to the value of the firstplural field.}
```

```latex
\glsentryfirstpluralaccess{⟨label⟩} \quad \text{glossaries-access}
\quad \text{Expands to the value of the firstpluralaccess field.}
```

```latex
\glsentryitem{⟨label⟩} \quad \text{glossaries v3.0+}
\quad \text{Increments and displays the glossaryentry counter, if appropriate.}
```

```latex
\Glsentrylong{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Displays the value of the long field without any formatting or indexing but with the first letter converted to upper case.}
```

```latex
\glsentrylong{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Expands to the value of the long field without any formatting or indexing.}
```

```latex
\glsentrylongaccess{⟨label⟩} \quad \text{glossaries-access}
\quad \text{Expands to the value of the longaccess field.}
```

```latex
\Glsentrylongpl{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Displays the value of the longplural field without any formatting or indexing but with the first letter converted to upper case.}
```

```latex
\glsentrylongpl{⟨label⟩} \quad \text{glossaries}^*
\quad \text{Expands to the value of the longplural field without any formatting or indexing.}
```

526
\glsentrylongpluralaccess{⟨label⟩} \hspace{1cm} glossaries-access
Expands to the value of the \texttt{longpluralaccess} field.

\Glsentryname{⟨label⟩} \hspace{1cm} glossaries*
Displays the value of the \texttt{name} field with the first character converted to upper case.

\glsentryname{⟨label⟩} \hspace{1cm} glossaries*
Expands to the value of the \texttt{name} field.

\Glsentryplural{⟨label⟩} \hspace{1cm} glossaries*
Displays the value of the \texttt{plural} field with the first letter converted to upper case.

\glsentryplural{⟨label⟩} \hspace{1cm} glossaries*
Expands to the value of the \texttt{plural} field.

\glsentrypluralaccess{⟨label⟩} \hspace{1cm} glossaries-access
Expands to the value of the \texttt{pluralaccess} field.

\Glsentryprefix{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefix} field with the first letter converted to upper case.

\glsentryprefix{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefix} field.

\Glsentryprefixfirst{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefixfirst} field with the first letter converted to upper case.

\glsentryprefixfirst{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefixfirst} field.

\Glsentryprefixfirstplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefixfirstplural} field with the first letter converted to upper case.

\glsentryprefixfirstplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefixfirstplural} field.

\Glsentryprefixplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefixplural} field with the first letter converted to upper case.

\glsentryprefixplural{⟨label⟩} \hspace{1cm} glossaries-prefix
Expands to the value of the \texttt{prefixplural} field.

\Glsentryshort{⟨label⟩} \hspace{1cm} glossaries*
Displays the value of the \texttt{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 \texttt{short} field without any formatting or indexing.

\glsentryshortaccess{⟨label⟩} \hspace{1cm} glossaries-access
Expands to the value of the \texttt{shortaccess} field.
\Glsentryshortpl\{\textit{label}\}\glossaries*
Displays the value of the \texttt{shortplural} field without any formatting or indexing but with the first letter converted to upper case.

\glsentryshortplural\{\textit{label}\}\glossaries*
Expands to the value of the \texttt{shortplural} field without any formatting or indexing.

\glsentryshortpluralaccess\{\textit{label}\}\glossaries-access
Expands to the value of the \texttt{shortpluralaccess} field.

\Glsentrysymbol\{\textit{label}\}\glossaries*
Displays the value of the \texttt{symbol} field with the first letter converted to upper case.

\glsentrysymbol\{\textit{label}\}\glossaries*
Expands to the value of the \texttt{symbol} field.

\glsentrysymbolaccess\{\textit{label}\}\glossaries-access
Expands to the value of the \texttt{symbolaccess} field.

\Glsentrysymbolplural\{\textit{label}\}\glossaries*
Displays the value of the \texttt{symbolplural} field with the first letter converted to upper case.

\glsentrysymbolplural\{\textit{label}\}\glossaries*
Expands to the value of the \texttt{symbolplural} field.

\glsentrysymbolpluralaccess\{\textit{label}\}\glossaries-access
Expands to the value of the \texttt{symbolpluralaccess} field.

\Glsentrytext\{\textit{label}\}\glossaries*
Displays the value of the \texttt{text} field with the first letter converted to upper case.

\glsentrytext\{\textit{label}\}\glossaries*
Expands to the value of the \texttt{text} field.

\glsentrytextaccess\{\textit{label}\}\glossaries-access
Expands to the value of the \texttt{textaccess} field.

\Glsentrytitlecase\{\textit{entry label}\}\{\textit{field label}\}\glossaries* v4.22+
Fetches the given field and applies \texttt{\textsc{capitalisewords}} to it.

\Glsentryuseri\{\textit{label}\}\glossaries*
Displays the value of the \texttt{user1} field with the first letter converted to upper case.

\glsentryuseri\{\textit{label}\}\glossaries*
Expands to the value of the \texttt{user1} field.

\Glsentryuserii\{\textit{label}\}\glossaries*
Displays the value of the \texttt{user2} field with the first letter converted to upper case.

\glsentryuserii\{\textit{label}\}\glossaries*
Expands to the value of the \texttt{user2} field.

\Glsentryuseriii\{\textit{label}\}\glossaries*
Displays the value of the \texttt{user3} field with the first letter converted to upper case.
General Command Summary

\glsentryuseriii{\langle label \rangle}
Expands to the value of the user3 field.

\Glsentryuseriv{\langle label \rangle}
Displays the value of the user4 field with the first letter converted to upper case.

\glsentryuseriv{\langle label \rangle}
Expands to the value of the user4 field.

\Glsentryuserv{\langle label \rangle}
Displays the value of the user5 field with the first letter converted to upper case.

\glsentryuserv{\langle label \rangle}
Expands to the value of the user5 field.

\Glsentryuservi{\langle label \rangle}
Displays the value of the user6 field with the first letter converted to upper case.

\glsentryuservi{\langle label \rangle}
Expands to the value of the user6 field.

\glsexpandfields
Switches on field expansion.

\glsextrapostnamehook{\langle label \rangle}
Additional category-independent code for the post-name hook.

\glsfielddef{\langle entry label \rangle}{\langle field label \rangle}{\langle definition \rangle}
Changes the value of the given entry’s field to \langle definition \rangle (localised by any scope).

\glsfieldedef{\langle entry label \rangle}{\langle field label \rangle}{\langle definition \rangle}
Changes the value of the given entry’s field to the full expansion of \langle definition \rangle (localised by any scope).

\glsfieldfetch{\langle label \rangle}{\langle field \rangle}{\langle cs \rangle}
Fetches the value of the given field for the given label and stores it in the command \langle cs \rangle.

\glsfieldgdef{\langle entry label \rangle}{\langle field label \rangle}{\langle definition \rangle}
Globally changes the value of the given entry’s field to \langle definition \rangle.

\glsfieldxdef{\langle entry label \rangle}{\langle field label \rangle}{\langle definition \rangle}
Globally changes the value of the given entry’s field to the full expansion of \langle definition \rangle.

\glsFindWidestLevelTwo{\langle glossary list \rangle}
Finds the widest name in the given glossaries for the top level and first two sub-levels.

\glsFindWidestTopLevelName{\langle glossary list \rangle}
CamelCase synonym for \glsFindWidestTopLevelName.

\glsfindwidesttoplevelname{\langle glossary list \rangle}
Finds the widest top-level name in the given glossaries.

\Glsfirst{\langle options \rangle}{\langle label \rangle}{\langle insert \rangle}
As \glsfirst but converts the first letter to upper case.
General Command Summary

\glsfirst[[options]]{[\langle label \rangle][\langle insert \rangle]}
glossaries  
Links to the entry’s definition in the glossary with the link text obtained from the \texttt{first} field without altering the first use flag.

\glsfirstabbrvdefaultfont{\langle text \rangle}
glossaries-extra  
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}
glossaries-extra v1.04+  
Used with “em” abbreviation styles to format the short form on first use. This defaults to \glsabbrvemfont.

\glsfirstabbrvhyphenfont{\langle text \rangle}
glossaries-extra v1.17+  
Used by the “hyphen” abbreviation styles to format the short form on first use.

\glsfirstabbrvonlyfont{\langle text \rangle}
glossaries-extra v1.17+  
Used with “only” abbreviation styles to format the short form on first use. The default definition just uses \glsabbrvonlyfont.

\glsfirstabbrvscfont{\langle text \rangle}
glossaries-extra v1.17+  
Used with “sc” abbreviation styles to format the short form on first use. This defaults to \glsabbrvscfont.

\glsfirstabbrvsmfont{\langle text \rangle}
glossaries-extra v1.17+  
Used with “sm” abbreviation styles to format the short form on first use. This defaults to \glsabbrvsmfont.

\glsfirstabbrvuserfont{\langle text \rangle}
glossaries-extra v1.04+  
Used with “user” abbreviation styles to format the short form on first use. The default definition just uses \glsabbrvuserfont.

\glsfirstaccessdisplay{\langle text \rangle}{\langle label \rangle}
glossaries-access  
Displays \langle text \rangle with the accessibility support provided by \texttt{glsentryfirstaccess}{\langle label \rangle}.

\glsfirstlongdefaultfont{\langle text \rangle}
glossaries-extra  
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}
glossaries-extra v1.04+  
Used with “long-em” abbreviation styles to format the long form on first use. This defaults to \glslongemfont.

\glsfirstlongfootnotefont{\langle text \rangle}
glossaries-extra v1.05+  
Used with the “footnote” abbreviation styles to format the long form on first use.

\glsfirstlonghyphenfont{\langle text \rangle}
glossaries-extra v1.17+  
Used by the “hyphen” abbreviation styles to format the long form on first use.

\glsfirstlongonlyfont{\langle text \rangle}
glossaries-extra v1.17+  
Used with “only” abbreviation styles to format the long form on first use. The default definition just uses \glslongonlyfont.
\glsfirstlonguserfont{(text)}
glossaries-extra v1.04+
Used with “user” abbreviation styles to format the long form on first use. The default
definition just uses \glslonguserfont.

\glsfirstpluralaccessdisplay{(text)}{(label)}
glossaries-access
Displays (text) 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 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 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 nogroupskip option).

\glshex
glossaries-extra v1.21+ (moved to glossaries-extra-bib2gls in v1.27)
Expands to \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 (true) if the category field for the entry given by (label) is (category).

\glsignore{(text)}
glossaries
Does nothing but when used as a location format 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.

\glsinlinedopostchild
glossary-inline v3.03+
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.

\glsinlinenamenameformat\{label\}|\{name\}
glossary-inline v3.03+
Format’s the entry’s name including target, if supported.

\glsinlineparentchildseparator
glossary-inline v3.03+
Separator between parent and child entries.

\glsinlinepostchild
glossary-inline v3.03+
Hook between child and next entry.

\glsinlineseparator
glossary-inline v3.03+
Separator between entries.

\glsinlinesubseparator
glossary-inline v3.03+
Separator between sub-entries.

\glslabel
glossaries
Only for use in the post-link hooks, this expands to the label of the entry that was last referenced.

\glslink|\{options\}|\{label\}|\{text\}
glossaries
Links to the entry’s definition in the glossary with the given link text without altering the first use flag.

Options:

\counter\{counter-name\}
glossaries
Sets the counter to use for the record

\format\{encap\}
glossaries
Sets the ENCAP for the record to \{encap\}, optionally with the start or end range markers

\hyper\{booleand\}
glossaries
Indicates whether or not to make a hyperlink to the relevant glossary entry

\hyperoutside\{booleand\}
glossaries-extra v1.21+
Determines whether \hyperlink should be outside of \glstextformat (default \hyperoutside=true) or inside (\hyperoutside=false)

\local\{booleand\}
glossaries
If true indicates to use \glslocalunset instead of the default global \glsunset to unset the first use flag

\noindex\{booleand\}
glossaries-extra
Indicates whether or not to suppress indexing
General Command Summary

prefix={{label}}
glossaries-extra v1.31+
Locally changes \glolinkprefix to the given \langle label\rangle.

textformat={{cs-name}}
glossaries-extra v1.30+
If set, replaces \glstextformat with the command given by the control sequence
name \langle cs-name\rangle to format the link text.

theHvalue={{value}}
glossaries-extra v1.19+
The hyperlink target corresponding to the value of \langle value\rangle, if appropriate.

thevalue={{value}}
glossaries-extra v1.19+
Overrides the record value so that it’s the given \langle value\rangle not obtained from the
associated counter.

wrgloss={{value}}
glossaries-extra v1.14+
Indicates whether to write the glossary information before \langle wrgloss=before\rangle or after
\langle wrgloss=after\rangle the link text (default: before).

 glireset{\langle label\rangle} 
glossaries
Locally resets the first use flag so that the entry is marked as not used.

 glsunset{\langle label\rangle} 
glossaries
Locally unsets the first use flag so that the entry is marked as having been used.

 gliaccessdisplay{\langle text\rangle}{\langle label\rangle} 
glossaries-access
Displays \langle text\rangle with the accessibility support provided by
\glsentrylongaccess{\langle label\rangle}.

 gliaccessdefaultfont{\langle text\rangle} 
glossaries-extra v1.04+
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.

 gliaccessemfont{\langle text\rangle} 
glossaries-extra v1.04+
Used with “long-em” abbreviation styles to format the long form using \emph.

 gliaccessextrasetwidest{\langle text\rangle} 
glossary-longextra v1.37+
Used with the styles provided by the glossary-longextra package to set the widest entry
name.

 gliaccessupdatewidest{\langle text\rangle} 
glossary-longextra v1.37+
As gliaccessextrasetwidest but only sets if \langle text\rangle is wider than the current value.

 gliaccessfont{\langle text\rangle} 
glossaries-extra v1.04+
Generic abbreviation font command for the long form.

 gliaccessfootnotefont{\langle text\rangle} 
glossaries-extra v1.05+
Used with the “footnote” abbreviation styles to format the long form.

 gliaccesshyphenfont{\langle text\rangle} 
glossaries-extra v1.17+
Used by the “hyphen” abbreviation styles to format the long form.

 gliaccessonlyfont{\langle text\rangle} 
glossaries-extra v1.17+
Used with “only” abbreviation styles to format the long form. The default definition just
uses gliaccessdefaultfont.
\glslongpluralaccessdisplay\{\text\}{\langle label\rangle}
glossaries-access
Displays \langle text\rangle with the accessibility support provided by \glsentrylongpluralaccess\{\langle label\rangle\}.
\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.
\glsname\{\langle options\rangle\}{\langle label\rangle}\{\langle insert\rangle\}
glossaries
Links to the entry’s definition in the glossary with the link text obtained from the \text{name} field without altering the first use flag.
\glsnameaccessdisplay\{\text\}{\langle label\rangle}
glossaries-access
Displays \langle text\rangle with the accessibility support provided by \glsentryaccess\{\langle label\rangle\}.
\glsnamefont\{\text\}
glossaries
Used by \glossentryname to format the name.
\glsnavhypertarget\{\langle type\rangle\}{\langle label\rangle}\{\text\}
glossary-hypernav
Creates a hyper target for the group given by \langle label\rangle for the given glossary type and uses \langle text\rangle for the hyperlink text.
\glsnodexpandfields
glossaries
Switches off field expansion.
\glsnoidxdisplayloc\{\langle prefix\rangle\}{\langle counter\rangle}\{\langle format\rangle\}{\langle location\rangle}
glossaries-v4.04+
Used to display a regular location in the \text{location} field (with a hyperlink, if enabled).
\glsnoidxloclist\{\langle location list cs\rangle\}
glossaries
Iterates over the given internal location list using the \glsnoidxloclisthandler handler.
\glsnoidxloclisthandler\{\langle location\rangle\}
glossaries
The handler used by the internal list loop function used in \glsnoidxloclist.
\glsnumberformat\{\text\}
glossaries
Default location format, uses \text{\glsnumber} if hyperlinks enabled otherwise just does \langle text\rangle.
\glsnumbersgroupname
glossaries
Language-sensitive name used for the numbers group and also used for the title of the glossary created with the \text{numbers} package option.
\glsopenbrace
glossaries*
Expands to a literal open brace \{ character.
\glspatchtabularx
glossaries
Preamble command that will patch the \text{tabularx} environment to deal with the problem of unsetting the first use flag either explicitly with \text{\glsunset} or implicitly through commands like \text{\gls} (does nothing if \text{tabularx} hasn’t been loaded).
General Command Summary

\glspersentchar \hfill glossaries*
Expands to a literal percent character %.

\GLSpl\{\langle\text{options}\rangle\}\{\langle\text{label}\rangle\}\{\langle\text{insert}\rangle\} \hfill glossaries
As GLS but shows the plural form.

\Glspl\{\langle\text{options}\rangle\}\{\langle\text{label}\rangle\}\{\langle\text{insert}\rangle\} \hfill glossaries
As \GLS but shows the plural form.

\glspl\{\langle\text{options}\rangle\}\{\langle\text{label}\rangle\}\{\langle\text{insert}\rangle\} \hfill glossaries
As \gls but shows the plural form.

\glspluralaccessdisplay\{\langle\text{text}\rangle\}\{\langle\text{label}\rangle\} \hfill glossaries-access
Displays \langle\text{text}\rangle with the accessibility support provided by \glsentrypluralaccess\{\langle\text{label}\rangle\}.

\glspluralsuffix \hfill glossaries*
The suffix used to construct the default plural.

\glspost-inline \hfill glosary-inline v3.03+
Glossary terminator.

\glspostdescription \hfill glossaries and modified by glossaries-extra
A hook added after the description in some glossary styles (all if the glossaries-extra-stylemods package is loaded to patch them). This hook is used to reflect the \nopostdot package option for \glossaries and the \postpunc option for \glossaries-extra.

\glsps\{\langle\text{label}\rangle\} \hfill glossaries-extra v1.07+
Shortcut for \glsxtrp\{short\}\{\langle\text{label}\rangle\}.

\glspt\{\langle\text{label}\rangle\} \hfill glossaries-extra v1.07+
Shortcut for \glsxtrp\{text\}\{\langle\text{label}\rangle\}.

\glsquote\{\langle\text{text}\rangle\} \hfill glossaries
Encapsulates \langle\text{text}\rangle with literal straight double-quotes "\langle\text{text}\rangle".

\glsrefentry\{\langle\text{label}\rangle\} \hfill glossaries v3.0+
When used with \entrycounter or \subentrycounter may be used to cross-reference the entry’s number in the glossary list with \ref.

\glsrenewcommand\{\langle\text{cs}\rangle\}\{\langle\text{n}\rangle\}\{\langle\text{def}\rangle\}\{\langle\text{code}\rangle\} \hfill glossaries-extra-bib2gls* v1.37+
Behaves like \renewcommand but only generates a warning rather than an error if the command isn’t already defined.

\glsreset\{\langle\text{label}\rangle\} \hfill glossaries
Resets the first use flag so that the entry is marked as not used.

\glsresetentrycounter \hfill glossaries
Resets the glossaryentry counter if the \entrycounter setting is on.

\glssee\{\langle\text{tag}\rangle\}\{\langle\text{label}\rangle\}\{\langle\text{xr label list}\rangle\} \hfill glossaries
Indexes a “see” cross-reference.
\glsseeformat{(tag)}{(labels)}{(location (ignored))}
glossaries
Formats the entries identified in the comma separated list of labels as a set of
cross-references, where each item in the list is encapsulated with \glsseeitem and each
element is separated with \glsseesep or \glsseelastsep.

\glsseeitem{(label)}
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{(label)}.

\glsseeitemformat{(label)}
glossaries v3.0+
Formats an element of the cross-reference list. With the base glossaries package this just
does \glsentrytext{(label)}. With glossaries-extra this uses either \glsentryshort or
\glsentryname depending on whether or not the short field has been set.

\glsseelastsep
glossaries
The separator used between the penultimate and ultimate entries of a cross-reference list.

\glsseesep
glossaries
The separator used between all but the last entries of a cross-reference list.

\glssetcategoryattribute{(category)}{(attribute)}{(value)}
glossaries-extra
Sets the value of the attribute for the given category.

\glssetexpandfield{(field)}
glossaries
Switches on field expansion for the given field.

\glssetnoexpandfield{(field)}
glossaries
Switches off field expansion for the given field.

\glssetwidest{(level)}{(text)}
glossary-tree
Used with the alttree style to set the widest entry name for the given level.

\glsshortaccessdisplay{(text)}{(label)}
glossaries-access
Displays \text{ (text)} with the accessibility support provided by
\glsentryshortaccess{(label)}.

\glsshortpluralaccessdisplay{(text)}{(label)}
glossaries-access
Displays \text{ (text)} with the accessibility support provided by
\glsentryshortpluralaccess{(label)}.

\glsshorttok
glossaries
Token register used in the construction of acronyms or abbreviations to allow the style
hooks to access the short form.

\glsshowtarget{(label)}
glossaries v4.32+
Used to show the target name when the debug={showtargets} option is on.

\glsstepentry{(label)}
glossaries
Increments the glossaryentry counter, which is defined with the entrycounter option,
and automatically labels it with \label.
General Command Summary

`\glssubentrycounterlabel`  
Governs the way the glossary subentry counter is displayed by `\glssubentryitem`.

`\glssubentryitem{⟨label⟩}`  
Increments and displays the glossary subentry counter, if appropriate.

`\glssymbol[(options)]{(label)⟨insert⟩}`  
Links to the entry's definition in the glossary with the link text obtained from the `symbol` field without altering the first use flag.

`\glssymbolaccessdisplay{(text)}{(label)}`  
Displays ⟨text⟩ with the accessibility support provided by `\glsentrysymbolaccess{(label)}`.

`\glssymbolpluralaccessdisplay{(text)}{(label)}`  
Displays ⟨text⟩ with the accessibility support provided by `\glsentrysymbolpluralaccess{(label)}`.

`\glssymbols grouplename`  
Language-sensitive name used for the symbols group and also used for the title of the glossary created with the `symbols` package option.

`\glstarget{(text)}{(label)}`  
Creates a hypertarget for the entry given by ⟨label⟩ (the target for commands like `\gls`) and displays ⟨text⟩.

`\Glstext[(options)]{(label)⟨insert⟩}`  
As `\Glstext` but converts the first letter to upper case.

`\Glstext[(options)]{(label)⟨insert⟩}`  
Links to the entry’s definition in the glossary with the link text obtained from the `text` field without altering the first use flag.

`\glstextaccessdisplay{(text)}{(label)}`  
Displays ⟨text⟩ with the accessibility support provided by `\glsentrytextaccess{(label)}`.

`\glstextformat{(text)}`  
Used by commands like `\gls` to format the link text.

`\glstextup{(text)}`  
Typesets ⟨text⟩ in an upright font (used to cancel the effect of `\textsc` in abbreviation styles that use `\glsabbrvscfont`).

`\glstildechar`  
Expands to a literal tilde ~ character.

`\glstreedefaultnamefmt{(text)}`  
Used as the default format for `\glstreenamefmt`, `\glstreegroupheaderfmt` and `\glstreenavigationfmt`.

`\glstreegroupheaderfmt{(text)}`  
Used with the tree styles to format the group headings.
General Command Summary

\glstreenamefmt\langle text\rangle \quad \text{glossary-tree v4.08+ and glossaries-extra-stylemods v1.31+}

Used with the tree styles to format the entry’s name.

\glstreenavigationfmt\langle text\rangle \quad \text{glossary-tree v4.22+ and glossaries-extra-stylemods v1.31+}

Used with the tree styles to format the navigation elements.

\glstreenonamedesc\langle label\rangle \quad \text{glossaries-extra-stylemods v1.31+}

Displays the pre-description separator, the description and the post-description hook for the treenoname styles.

\glstreepredesc\langle label\rangle \quad \text{glossary-tree v4.26+}

Separator used before the description for the tree styles.

\glstreeprelocation \quad \text{glossaries-extra-stylemods v1.21+}

Inserted before the location list for top-level entries in the tree-like styles.

\glstriggerrecordformat\langle text\rangle \quad \text{glossaries-extra v1.21+}

Does nothing but when used as a location format \text{bib2gls} recognises it as an ignored record indexed by commands like \text{\rgls}.

\glsunset\langle label\rangle \quad \text{glossaries}

Unsets the first use flag so that the entry is marked as having been used.

\glsetwidest\langle level\rangle\langle text\rangle \quad \text{glossaries-extra-stylemods v1.23+}

As \text{\glssetwidest} but only sets if \langle text\rangle is wider than the current value.

\glsuseabbrvfont\langle text\rangle\langle category\rangle \quad \text{glossaries-extra v1.21+}

Applies the formatting command used for the short form for the abbreviation style associated with the given category.

\glsuselongfont\langle text\rangle\langle category\rangle \quad \text{glossaries-extra v1.21+}

Applies the formatting command used for the long form for the abbreviation style associated with the given category.

\glsuserdescription\langle description\rangle\langle label\rangle \quad \text{glossaries-extra v1.30+}

Used with “user” abbreviation styles to encapsulate the description. Just does \text{\glslonguserfont\langle description\rangle} by default.

\glsuseri\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 \text{user1} field without altering the first use flag.

\glsuserii\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 \text{user2} field without altering the first use flag.

\glsuseriii\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 \text{user3} field without altering the first use flag.

\glsuseriv\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 \text{user4} field without altering the first use flag.
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.

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.

This command is written to the .aux file each time an entry is indexed to provide \texttt{bib2gls} with the record information.

Used instead of \texttt{glsxtr@record} when the \texttt{record=\nameref} option is used.

This internal command is written to the .aux file by \texttt{\glsxtrresourcem} to provide \texttt{bib2gls} with the resource information.

This command simply expands to \(n\), the value of the \texttt{wrglossary} counter for the given page.

Used by commands like \texttt{\gls} to format the link text for (non-regular) abbreviations.

Used with the “footnote” abbreviation styles to do the footnote. The \(\texttt{label}\) is ignored by default. The \(\texttt{long form}\) includes the font changing command. This just does \texttt{\footnote{\texttt{long form}}}.

The default suffix used to construct the plural for the short form of abbreviations. This just uses \texttt{\glspluralsuffix}. If you don’t want a plural suffix, you can use the \texttt{noshortplural} attribute.

Expands to the default glossary type when using \texttt{\newabbreviation}.

Appends \(\texttt{prefix}\) to the prefix label list.

Used by the alttree styles to indicate a paragraph break that retains the hanging indent.

Used by the alttree styles to format the symbol, description and location.

Makes commands like \texttt{\gls} and \texttt{\glslink} (but not \texttt{\glsadd}) automatically insert \texttt{\glsadd} if the format (supplied in the optional argument of the
invoking \gls, \glslink etc) matches any in the given comma-separated elements of ⟨format list⟩. The format isn’t automatically applied to the \glsadd options.

\glsxtrautoindexassignsort{⟨cs⟩}{⟨label⟩} glossaries-extra v1.16+
Assigns the sort value for \index when using auto-indexing.

\glsxtrautoindexentry{⟨label⟩} glossaries-extra v1.16+
Used for the actual value in \index when using auto-indexing.

\texttt{\textbackslash GlsXtrBibTeXEntryAliases} glossaries-extra-bib2gls v1.29+
Expands the set of common entry aliases for \texttt{@bibtexentry}.

\glsxtrbookindexname{⟨label⟩} glossary-bookindex
Used with the bookindex style to format the entry’s name.

\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⟩.

\texttt{\textbackslash GlsXtrDefaultResourceOptions} glossaries-extra v1.40+
Provides default options for \glsxtrresourcefile.

\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).

\texttt{\textbackslash GlsXtrDualBackLink}{⟨text⟩}{⟨label⟩} glossaries-extra-bib2gls v1.30+
Creates a hyperlink to the dual entry whose label is obtained from the field given by \texttt{\textbackslash GlsXtrDualField}. 

540
General Command Summary

\GlsXtrDualField\textgls{glossaries-extra-bib2gls v1.30+}
The field used to store the dual label. This defaults to \texttt{dual} but will need to be redefined if a different value is given by \texttt{dual-field}.

\glsxtremsuffix\textgls{glossaries-extra}
The suffix used to construct the plural for the short form of abbreviations with the “em” styles. This defaults to \texttt{\glsxtrabbrvpluralsuffix}.

\GlsXtrEnableInitialTagging\{\langle category list\rangle\}\{\langle cs\rangle\}\textgls{glossaries-extra *}
Defines the control sequence \texttt{\langle cs\rangle} to be used with abbreviation tagging with the given categories.

\glsxtrenablerecordcount\textgls{glossaries-extra v1.21+}
Redefines \texttt{\gls} etc to their \texttt{\rgls} counterpart.

\glsxtrrendfor\textgls{glossaries-extra v1.24+}
May be used within the handler macro of \texttt{\glsxtrforcsvfield} to prematurely break the loop.

\glsxtrentryfmt\{\langle label\rangle\}\{\langle text\rangle\}\textgls{glossaries-extra v1.12+}
Alternative to \texttt{\glsxtrfmt} for use in section headings.

\glsxtrentryparentname\{\langle label\rangle\}\textgls{glossaries-extra * v1.39+}
Expands to the entry’s parent’s name.

\GlsXtrExpandedFmt\{\langle cs\rangle\}\{\langle text\rangle\}\textgls{glossaries-extra v1.30+}
Fully expands \texttt{\langle text\rangle} and then does \texttt{\langle cs\rangle\{expanded text\}}.

\glsxtrfielddolistloop\{\langle label\rangle\}\{\langle field\rangle\}\textgls{glossaries-extra v1.12+}
Iterates over the items the given field, which contains an etoolbox internal list.

\glsxtrfieldforlistloop\{\langle label\rangle\}\{\langle field\rangle\}\{\langle handler\rangle\}\textgls{glossaries-extra v1.29+}
Iterates over the items the given field, which contains an etoolbox internal list, using the given handler.

\glsxtrfieldformatcsvlist\{\langle label\rangle\}\{\langle field\rangle\}\textgls{glossaries-extra v1.42+}
Formats the items in the given field, which contains a comma-separated list, using the same handler as \texttt{\DTLformatlist}.

\glsxtrfieldformatlist\{\langle label\rangle\}\{\langle field\rangle\}\textgls{glossaries-extra v1.42+}
Formats the items in the given field, which contains an etoolbox internal list, using the same handler as \texttt{\DTLformatlist}.

\glsxtrfieldifinlist\{\langle label\rangle\}\{\langle field\rangle\}\{\langle item\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}\textgls{glossaries-extra v1.12+}
Tests if the given item is in the given field that contains an etoolbox internal list.

\glsxtrfieldlistadd\{\langle label\rangle\}\{\langle field\rangle\}\{\langle item\rangle\}\textgls{glossaries-extra v1.12+}
Adds the given item to the given field that contains an etoolbox internal list.

\glsxtrfieldxifinlist\{\langle label\rangle\}\{\langle field\rangle\}\{\langle item\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}\textgls{glossaries-extra v1.12+}
Tests if the expansion of the given item is in the given field that contains an etoolbox internal list.
\glsxtrfmt\[\langle\text{options}\rangle\]\[\langle\text{label}\rangle\]\[\langle\text{text}\rangle\]
glossaries-extra v1.12+

Formats the given text according to the formatting command identified by the value of the field obtained from \GlsXtrFmtField.

\glsxtrfmt*\[\langle\text{options}\rangle\]\[\langle\text{label}\rangle\]\[\langle\text{text}\rangle\]\[\langle\text{insert}\rangle\]
glossaries-extra v1.23+

Like \glsxtrfmt but inserts extra material into the link text but outside of the formatting command.

\GlsXtrFmtDefaultOptions
glossaries-extra v1.12+

The default options used by \glsxtrfmt.

\glsxtrfmtdisplay\{\langle\cs-name\rangle\}\{\langle\text\rangle\}\{\langle\text\rangle\}
glossaries-extra

Used by \glsxtrfmt to format the given \langle\text\rangle where \langle\cs-name\rangle is obtained from the field identified by \GlsXtrFmtField and \langle\text\rangle 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 name field is assigned to when the abbreviation is defined with \newabbreviation (defaults to the short form).

\glsxtrforcsvfield\{\langle\text\rangle\}\{\langle\cs-name\rangle\}\{\langle\text\rangle\}\{\langle\handler\rangle\}\{\langle\element\rangle\}
glossaries-extra v1.24+

Iterates over the comma-separated list in the given \langle\cs-name\rangle for the entry identified by \langle\label\rangle and performs \langle\handler\rangle\{\langle\element\rangle\} on each element of the list, where \langle\handler\rangle is a control sequence which takes a single argument.

\GlsXtrForeignText\[\langle\text\rangle\]
glossaries-extra v1.32+

Encapsulates \langle\text\rangle 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\{\langle\cs\rangle\}
glossaries-extra v1.31+

Iterates over all the entry whose labels are stored in the buffer that was started with \GlsXtrStartUnsetBuffering and implements \langle\cs\rangle\{\langle\label\rangle\} at each iteration.

\glsxtrfractionrules
glossaries-extra-bib2gls v1.27+

Collation sub-rule for vulgar fraction characters.

\glsxtrfull\[\langle\text\rangle\]
glossaries-extra

Links to the entry’s definition in the glossary with the link text obtained from the long and short fields (using the appropriate abbreviation style) without altering the first use flag.

\glsxtrfullsep\{\langle\text\rangle\}
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.
General Command Summary

\glsxtrGeneralLatinIIIrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIrules but includes
\(\text{Ð}/\text{ð}\) between \(\text{D}/\text{d}\) and \(\text{E}/\text{e}\) and \(\beta\) treated as “sz”).

\glsxtrGeneralLatinIIrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIrules but includes
\(\text{Ð}/\text{ð}\) between \(\text{D}/\text{d}\) and \(\text{E}/\text{e}\) and \(\beta\) treated as “ss”).

\glsxtrGeneralLatinIrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for Latin characters (basic Latin set plus subscript and superscript
Latin characters).

\glsxtrGeneralLatinIVrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIrules but includes
\(\text{Ð}/\text{ð}\) between \(\text{D}/\text{d}\) and \(\text{E}/\text{e}\) and \(\text{Æ}/\text{æ}\) treated as AE/ae, Œ/œ treated as OE/oe,
\(\text{Þ}/\text{þ}\) treated as TH/th and \(\beta\) treated as “ss”).

\glsxtrGeneralLatinVIIIrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for Latin characters: as \glsxtrGeneralLatinIrules but includes
\(\text{Æ}/\text{æ}\) treated as A/a, Œ/œ treated as OE/oe, \(\text{Þ}/\text{þ}\) treated as TH/th, \(\beta\) treated as “ss”, \(\text{Ð}/\text{ð}\)
treated as D/d, Ø/ø treated as O/o and Ł/ł treated as L/l.

\glsxtrGeneralLatinVIIrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for Latin characters: as \glsxtrGeneralLatinIrules but includes
\(\text{Æ}/\text{æ}\) between A/a and B/b, \(\text{Ð}/\text{ð}\) between D/d and E/e, \(\delta/\gamma\) (insular G) as G/g, Œ/œ between
O/o and P/p, \(\text{f}\) (long S) equivalent to S/s, \(\text{þ}/\text{þ}\) between T/t and U/u and \(\text{ƿ}/\text{ƿ}\) (wynn) as W/w.

\glsxtrGeneralLatinVIrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for Latin characters (as \glsxtrGeneralLatinIrules but includes
\(\text{Ñ}/\text{ñ}\) between D/d and E/e and \(\beta\) treated as “ss”).

\glsxtrgeneralpuncrules\textsuperscript{glossaries-extra-bib2gls v1.27+}
Collation sub-rule for general punctuation characters.

\glsxtrglossentry{⟨label⟩}\textsuperscript{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{⟨header⟩}{⟨label⟩}{⟨field⟩}\textsuperscript{glossaries-extra v1.22+}
Like \glsxtrglossentry but uses the value given in the supplied internal ⟨field⟩ where
⟨header⟩ is the code to use in the header (leave empty for default).

\glsxtrgroupfield\textsuperscript{glossaries-extra v1.21+}
Expands to the field label used to store the entry group labels.

\GLSXTRhiername{⟨label⟩}\textsuperscript{glossaries-extra v1.37+}
Displays the hierarchical name for the entry identified by ⟨label⟩ with each level
separated by \glsxtrhiernamesep where each name is 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 is 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 each name has the first letter converted to upper case.

\Glsxthiernamesep
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+
Displays the hierarchical name for the entry identified by ⟨label⟩ with each level separated by \glsxtrhiernamesep.

\glsxtrhiernamesep
glossaries-extra v1.37+
Separator between hierarchical levels displayed with \glsxthiernamesep (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.

\glsxtrifcustomdiscardperiod{⟨true⟩}{⟨false⟩}
glossaries-extra v1.23+
Should expand to ⟨true⟩ if the post-link hook should check for a following full stop (in addition to attribute checks) otherwise should expand to ⟨false⟩.

\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{{\text{field label}}}{{\text{entry label}}}{{\text{text}}}{{true}}{{false}} \quad \text{glossaries-extra v1.21+}
Tests if the given field value is the same as \text{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{{\text{field label}}}{{\text{entry label}}}{{\text{text}}}{{true}}{{false}} \quad \text{glossaries-extra v1.31+}
Like \GlsXtrIfFieldEqStr but first (protected) fully expands \text{text} (but not the field value). The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIfFieldNonZero{{\text{field}}}{{\text{entry label}}}{{true}}{{false}} \quad \text{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 \text{true} or \text{false} with \text{glscurrentfieldvalue}. The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIfFieldUndef{{\text{field label}}}{{\text{entry label}}}{{true}}{{false}} \quad \text{glossaries-extra v1.23+}
Tests if the given field isn’t defined for the given entry, which may also not exist.

\glsxtrifhasfield{{\text{field label}}}{{\text{entry label}}}{{true}}{{false}} \quad \text{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 \text{true} and \text{false}.

\glsxtrifhasfield*{{\text{field label}}}{{\text{entry label}}}{{true}}{{false}} \quad \text{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.

\GlsXtrIfHasNonZeroChildCount{{\text{entry label}}}{{true}}{{false}} \quad \text{glossaries-extra-bib2gls v1.31+}
For use with the save-child-count resource option, this uses \GlsXtrIfFieldNonZero to test if the \text{childcount} field has a non-zero value. The value can be referenced in \text{true} or \text{false} with \text{glscurrentfieldvalue}.

\glsxtrifhyphenstart{{\text{text}}}{{true}}{{false}} \quad \text{glossaries-extra v1.17+}
Used by the “hyphen” abbreviation styles, this checks if \text{text} starts with a hyphen.

\glsxtrifinmark{{true}}{{true}} \quad \text{glossaries-extra v1.07+}
Used by commands like \glsfmtshort, this expands to \text{true} in page headings and the table of contents, otherwise it expands to \text{false}.

\glsxtriflabelinlist{{\text{label}}}{{\text{list}}}{{true}}{{false}} \quad \text{glossaries-extra v1.21+}
Tests if the \text{label} is contained in the comma-separated \text{list}, where both \text{label} and \text{list} are fully expanded before testing. This test is designed for \text{labels} that are fully expandable.

\GlsXtrIfUnusedOrUndefined{{\text{label}}}{{true}}{{false}} \quad \text{glossaries-extra v1.34+}
Does \text{true} if the entry given by \text{label} hasn’t been used or is undefined, otherwise it does \text{false}. This command is not for use in the post-link hooks.
\glsxtrifwasfirstuse\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries-extra}

Only for use in the post-link hooks this tests if the entry just referenced was used for the first time.

\GlsXtrIfXpFieldEqXpStr\{⟨field label⟩\}\{⟨entry label⟩\}\{⟨text⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries-extra v1.31+}

Like \GlsXtrIfFieldEqStr but first (protected) fully expands both the field value and ⟨text⟩. The unstarred form adds implicit grouping. The starred form (new to v1.39) doesn’t.

\GlsXtrIndexCounterLink\{⟨text⟩\}\{⟨label⟩\} \textit{glossaries-extra-bib2gls v1.29+}

Creates a hyperlink to the wrglossary location obtained from the indexcounter field.

\glsxtrindexseealso\{⟨label⟩\}\{⟨xr list⟩\} \textit{glossaries-extra v1.16+}

Indexes a “see also” cross-reference.

\glsxtrinsertinsidefalse \textit{glossaries-extra v1.02+}

Sets the \ifglsxtrinsertinside switch to false.

\glsxtrinsertsidetrue \textit{glossaries-extra v1.02+}

Sets the \ifglsxtrinsertinside switch to true.

\glsxtrLatinAA \textit{glossaries-extra-bib2gls v1.27+}

Collation sub-rule for Å/å.

\glsxtrLatinOslash \textit{glossaries-extra-bib2gls v1.27+}

Collation sub-rule for Ø/ø.

\GlsXtrLoadResources\[⟨options⟩\] \textit{glossaries-extra v1.11+}

A shortcut command that uses \glsxtrresourcefile.

\glsxtrlocalsetgrouptitle\{⟨group label⟩\}\{⟨group title⟩\} \textit{glossaries-extra v1.24+}

Locally sets the title for the group identified by the given label.

\GlsXtrLocationField \textit{glossaries-extra v1.37+}

Expands to the internal name of the field storing the location list, defaulting to location.

\glsxtrlocationhyperlink\{⟨counter⟩\}\{⟨prefix⟩\}\{⟨location⟩\} \textit{glossaries-extra 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\[⟨options⟩\]\{⟨label⟩\} \textit{glossaries-extra}

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\{⟨long⟩\}\{⟨label⟩\}\{⟨insert⟩\} \textit{glossaries-extra v1.17+}

Used by the long-hyphen-postshort-hyphen abbreviation to format the long form and check if the ⟨insert⟩ starts with a hyphen.

\glsxtrlonghyphennohref\{⟨label⟩\}\{⟨long⟩\}\{⟨insert⟩\} \textit{glossaries-extra 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.
General Command Summary

\glsxtrlonghyphenshort{⟨label⟩}{⟨long⟩}{⟨short⟩}{⟨insert⟩}
glossaries-extra 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 name field is assigned to when the abbreviation is defined with \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 name field is assigned to when the abbreviation is defined with \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 name field is assigned to when the abbreviation is defined with \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 name field is assigned to when the abbreviation is defined with \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 name field is assigned to when the abbreviation is defined with \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).

\glsxtrmultisupplocation{⟨location⟩}{⟨src⟩}{⟨format⟩}
glossaries-extra-bib2gls v1.36+
Used by \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 \textit{glossaries-extra v1.22+}
Suppresses the post-description punctuation without suppressing the post-description hook.

\glsxtronelydescname \textit{glossaries-extra v1.17+}
Hook for the long-only-short-only-desc style that expands to the value that the \textit{name} field is assigned to when the abbreviation is defined with \texttt{\newabbreviation} (defaults to the long form).

\glsxtrononyme \textit{glossaries-extra v1.25+}
Hook for the long-only-short-only style that expands to the value that the \textit{name} field is assigned to when the abbreviation is defined with \texttt{\newabbreviation} (defaults to the short form).

\glsxtronysuffix \textit{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 \texttt{\glsxtrabbbvpluralsuffix}.

\glsxtrp{\langle field \rangle}{\langle label \rangle} \textit{glossaries-extra v1.07+}
Displays the given \textit{field} value for the entry given by \textit{label} (no hyperlinks, except in the glossary, and no indexing by default, but includes formatting, if appropriate).

\glsxtrpageref{\langle label \rangle} \textit{glossaries-extra v1.11}
When used with \texttt{\entrycounter} or \texttt{\subentrycounter} may be used to cross-reference the entry’s number in the glossary list with \texttt{\pageref}.

\glsxtrparen{\textit{text}} \textit{glossaries-extra v1.17+}
Used to markup parenthetical material, such as in \texttt{\glsxtrpostlinkAddDescOnFirstUse} or in the long-short and short-long abbreviation styles.

\glsxtrpostdescabbreviation \textit{glossaries-extra}
Hook used after the description is displayed in the glossary for entries that have the \textit{category} set to \texttt{abbreviation}.

\glsxtrpostdesc{\textit{category}} \textit{glossaries-extra}
Hook used after the description is displayed in the glossary for entries that have the \textit{category} set to \textit{category}. Common category hooks such as \texttt{\glsxtrpostdescgeneral} are provided by \texttt{glossaries-extra}. If required, this hook can be defined with \texttt{\glsdefpostdesc}.

\glsxtrpostdescgeneral \textit{glossaries-extra}
Hook used after the description is displayed in the glossary for entries that have the \textit{category} set to \texttt{general}.

\glsxtrpostdescsymbol \textit{glossaries-extra}
Hook used after the description is displayed in the glossary for entries that have the \textit{category} set to \texttt{symbol}.

\glsxtrposthyphenlong{\langle label \rangle}{\langle insert \rangle} \textit{glossaries-extra v1.17+}
Used by the “postlong-hyphen” styles in the post-link hook.
\glsxtrposthyphenshort\{\langle label \rangle\}\{\langle insert \rangle\}  
  \texttt{glossaries-extra v1.17+}
  Used by the long-hyphen-postshort-hyphen style in the post-link hook.

\glsxtrposthyphenshortsubsequent\{\langle label \rangle\}\{\langle insert \rangle\}  
  \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
  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 symbol
  field (if set) followed by the value of the description field.

\glsxtrpostlinkAddSymbolOnFirstUse  \texttt{glossaries-extra}
  Only for use in the post-link hooks, this appends a space and the value of the
  symbol field in parentheses if the entry that was just referenced was used for the first time
  and has the symbol field set.

\glsxtrpostlink\{\langle category \rangle\}  \texttt{glossaries-extra}
  Hook used after commands like \texttt{\gls} for entries that have the category set to
  \{\langle category \rangle\}. If required, this hook can be defined with \texttt{\glsdefpostlink}.

\glsxtrpostname\{\langle category \rangle\}  \texttt{glossaries-extra}
  Hook used by \texttt{\glossentryname} for entries that have the category set to
  \{\langle category \rangle\}. If required, this hook can be defined with \texttt{\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\{\langle prefix \rangle\}  \texttt{glossaries-extra-bib2gls v1.37+}
  Prepends \{\langle prefix \rangle\} to the prefix label list.

\GlsXtrProvideBibTeXFields  \texttt{glossaries-extra-bib2gls v1.29+}
  Defines the standard Bib\TeX\ fields using \texttt{\glsaddstoragekey}.

\glsxtrprovidecommand\{\langle cs \rangle\}\{\langle n \rangle\}\{\langle def \rangle\}\{\langle code \rangle\}  \texttt{glossaries-extra-bib2gls v1.27+}
  Behaves like \texttt{\providecommand} in the document but like \texttt{\renewcommand} in bib2gls.

\glsxtrprovidestoragekey\{\langle key \rangle\}\{\langle default value \rangle\}\{\langle no link cs \rangle\}  \texttt{glossaries-extra v1.12+}
  Adds a new key, if not already defined, for use in \texttt{\newglossaryentry} and an associated
  command to access it where (unlike \texttt{\glsaddstoragekey}) the \{\langle no link cs \rangle\} part may be
  empty if unrequired.

\glsxtrregularfont\{\langle text \rangle\}  \texttt{glossaries-extra v1.04+}
  Used by commands like \texttt{\gls} to format the link text for regular terms.
General Command Summary

\glxtrresourcefile\{(options)\}\{(filename)\}  \quad \text{glossaries-extra v1.08+}
Input the .glstex file created by bib2gls and write resource instructions to the .aux file.

\glxtrresourceinit  \quad \text{glossaries-extra v1.21+}
Provides code that locally redefines commands during the protected write operation performed by \glxtrresourcefile.

\glxtrrestorepostpunc  \quad \text{glossaries-extra v1.23+}
Used within post-description category hooks, this restores the post-description punctuation if it’s been suppressed with \glxtrnopostpunc.

\glxtrRevertTocMarks  \quad \text{glossaries-extra v1.07+}
Restores original behaviour of \tableofcontents so that \glsxtrifinmark expands to \langle false \rangle in the table of contents.

\glxtrscsuffix  \quad \text{glossaries-extra}
The suffix used to construct the plural for the short form of abbreviations with the small-cap “sc” styles. This counteracts the effect of \textsc using \glstextup.

\glxtrseelist\{\langle xr label list \rangle\}  \quad \text{glossaries-extra v1.16+}
Formats the list of cross-reference labels, without the initial “see” tag.

\glxtrsetaliasnoindex  \quad \text{glossaries-extra v1.12+}
Hooks into the alias noindex setting.

\GlsXtrSetDefaultGlsOpts\{\langle options \rangle\}  \quad \text{glossaries-extra}
Set the default options for commands like \gls.

\GlsXtrSetDefaultNumberFormat\{\langle format \rangle\}  \quad \text{glossaries-extra v1.19+}
Set the default format to use if the format key isn’t set.

\GlsXtrSetField\{\langle entry label \rangle\}\{\langle field label \rangle\}\{\langle value \rangle\}  \quad \text{glossaries-extra v1.12+}
Assigns the given \langle value \rangle to the field identified by \langle field label \rangle for the entry identified by \langle entry label \rangle.

\glxtrsetglossarylabel\{\langle label \rangle\}  \quad \text{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.

\glxtrsetgrouptitle\{\langle group label \rangle\}\{\langle group title \rangle\}  \quad \text{glossaries-extra v1.14+}
Globally sets the title for the group identified by the given label.

\glxtrsetpopts\{\langle options \rangle\}  \quad \text{glossaries-extra v1.07+}
Sets the default options for \glxtrp.

\GlsXtrSetRecordCountAttribute\{\langle category list \rangle\}\{\langle value \rangle\}  \quad \text{glossaries-extra v1.21+}
Sets the recordcount attribute to \langle value \rangle for the given categories.

550
\glsxtrSetWidest\{\langle type\rangle}\{\langle level\rangle}\{\langle text\rangle\} \quad \text{glossaries-extra-bib2gls v1.37+}

Used by \bibglssetwidest to set the widest entry name for the given level for the alttree style and the styles provided by glossary-longextra.

\glsxtrSetWidestFallback\{\langle max depth\rangle\}\{\langle list\rangle\} \quad \text{glossaries-extra-bib2gls v1.37+}

Used by \bibglssetwidesttoplevelfallback and \bibglssetwidestfallback to set the widest entry name for the alttree style and the styles provided by glossary-longextra using the commands provided by glossaries-extra-stylemods.

\glsxtrshort\{\langle options\rangle\}\{\langle label\rangle\} \quad \text{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 \quad \text{glossaries-extra v1.17+}

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\} \quad \text{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\} \quad \text{glossaries-extra v1.17+}

Used by the short-hyphen-long-hyphen style to format the full form.

\glsxtrshortlongdescname \quad \text{glossaries-extra v1.17+}

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 \quad \text{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 \quad \text{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 \quad \text{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).

\glsxtrsm_suffix \quad \text{glossaries-extra}

The suffix used to construct the plural for the short form of abbreviations with the "sm" styles. This defaults to \glsxtrabbrvpluralsuffix.

\glsxtrspacerules \quad \text{glossaries-extra-bib2gls v1.27+}

Collation sub-rule for space characters.
\GlsXtrStandaloneEntryName{⟨label⟩} \quad \text{glossaries-extra v1.37+}

Used within \glsxtrglossentry to display the name (with a hypertarget, if supported).

\GlsXtrStandaloneEntryOther{⟨label⟩}{⟨field⟩} \quad \text{glossaries-extra v1.37+}

Used within \glsxtrglossentryother to display the given field value (with a hypertarget, if supported).

\GlsXtrStandaloneGlossaryType \quad \text{glossaries-extra v1.31+}

Expands to the label for \currentglossary within \glsxtrglossentry and \glsxtrglossentryother.

\GlsXtrStandaloneSubEntryItem{⟨label⟩} \quad \text{glossaries-extra v1.31+}

Used within \glsxtrglossentry and \glsxtrglossentryother to display sub-item labels.

\GlsXtrStartUnsetBuffering \quad \text{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 \quad \text{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⟩} \quad \text{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.

\glsxtrsupplocationurl \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.

\glsxtrsrunstodo{⟨label⟩} \quad \text{glossaries-extra v1.12+}

Displays the entry given by ⟨label⟩ using \glossentry or \subglossentry depending on the entry’s hierarchical level (taking \text{leveloffset} into account).

\GLSxtrusefield{⟨entry label⟩}{⟨field label⟩} \quad \text{glossaries-extra* v1.37+}

As \glsxtrusefield but converts the value to upper case.

\GLSxtrusefield{⟨entry label⟩}{⟨field label⟩} \quad \text{glossaries-extra* v1.12+}

Like \glsxtrusefield but converts the first letter to upper case.

\GLSxtrusefield{⟨entry label⟩}{⟨field label⟩} \quad \text{glossaries-extra* v1.12+}

Expands to the value of the given field for the given entry.
General Command Summary

\texttt{\textbackslash glsxtruserfield} \textit{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 \texttt{useri}.

\texttt{\textbackslash glsxtruserparen\{\text\}(\text{label})\}} \textit{glossaries-extra v1.04+}

Used by the “user” abbreviation styles to format the parenthetical material where \texttt{(text)} is the default parenthetical text and \texttt{(label)} is the entry’s label. This checks the field given by \texttt{\textbackslash glsxtruserfield} and, if set, the \texttt{(text)} is followed by a comma and the user value.

\texttt{\textbackslash glsxtrusersuffix} \textit{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 \texttt{\textbackslash glsxtrabbrvpluralsuffix}.

\texttt{\textbackslash glsxtrusesee\{\text\}\}} \textit{glossaries-extra v1.06+}

Applies \texttt{\textbackslash glsseeformat} to the entry’s \texttt{see} field if not empty.

\texttt{\textbackslash glsxtruseseealso\{\text\}\}} \textit{glossaries-extra v1.16+}

Applies \texttt{\textbackslash glsseeformat} to the entry’s \texttt{seealso} field if not empty.

\texttt{\textbackslash glsxtruseseealsoformat\{\text{xr list}\}} \textit{glossaries-extra v1.16+}

Used to format the entries whose labels are given in \texttt{\{xr list\}} as a list of “see also” cross-references.

\texttt{\textbackslash glsxtruseseeformat\{\text\}\}} \textit{glossaries}

Formats the entries identified in the comma separated list of labels as a set of cross-references.

\texttt{\textbackslash glsxtrword\{\text\}} \textit{glossaries-extra v1.17+}

Used to encapsulate each word in the long form of an abbreviation by the \texttt{markwords} attribute.

\texttt{\textbackslash glsxtrwordsep} \textit{glossaries-extra v1.17+}

Used to mark spaces between each word in the long form of an abbreviation by the \texttt{markwords} attribute.

\texttt{\textbackslash heartsuit} \textit{kernel command* (maths mode)}

Heart symbol (♥).

\texttt{\hyperbf\{\text\}} \textit{glossaries}

A location format that uses the bold font that also has a hyperlink (if enabled).

\texttt{\hyperemph\{\text\}} \textit{glossaries}

A location format that uses \texttt{\emph} to set the font and also has a hyperlink (if enabled).

\texttt{\hyperit\{\text\}} \textit{glossaries}

A location format that uses the italic font that also has a hyperlink (if enabled).

\texttt{\hyperlink\{\text{target name}\}\{\text\}} \textit{hyperref*}

Create a hyperlink to \texttt{\{target name\}} with the given \texttt{\{text\}}.

553
General Command Summary

\hyperref
This command has 2 forms:

\hyperref{⟨URL⟩}{⟨category⟩}{⟨name⟩}{⟨text⟩}
Create a hyperlink to an external location with the anchor formed from
⟨category⟩.⟨name⟩ and the displayed ⟨text⟩.

\hyperref{⟨label⟩}{⟨text⟩}
Create an internal hyperlink with the displayed ⟨text⟩ to the same place that
\ref{⟨label⟩} would be linked. Note that the ⟨label⟩ argument isn’t optional. The
square bracket notation disambiguates from the syntax for the external form of
\hyperref.

\hyperrm{⟨text⟩}
glossaries
A location format that uses the serif (Roman) font that also has a hyperlink (if enabled).

\hypersf{⟨text⟩}
glossaries
A location format that uses the sans-serif font that also has a hyperlink (if enabled).

I

\ifcase ⟨number⟩ \TeX primitive*
Case conditional.

\ifcsdef{⟨cs-name⟩}{⟨true⟩}{⟨false⟩}
etoolbox
Tests if the control sequence given by ⟨cs-name⟩ is defined.

\ifcsstrequal{⟨cs-name1⟩}{⟨cs-name2⟩}{⟨true⟩}{⟨false⟩}
etoolbox
Tests if the replacement text of the command given by the control sequence name
⟨cs-name1⟩ equals the replacement text of the command given by the control sequence
name ⟨cs-name2⟩.

\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+
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⟩\} \textit{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⟩\} \textit{glossaries v4.46+ (or glossaries-extra v1.44+)}
The starred form of \ifglossaryexists treats ignored glossaries as existing.

\ifglsentryexists\{⟨label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries}
Tests if the entry given by ⟨label⟩ exists.

\ifglsfieldcseq\{⟨entry label⟩\}\{⟨field label⟩\}\{⟨cs-name⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{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⟩\} \textit{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.

\ifglsfieldeq\{⟨entry label⟩\}\{⟨field label⟩\}\{⟨string⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{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).

\ifglshaschildren\{⟨entry label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{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 label⟩ as the value of the parent field. With \bib2gls, a more efficient approach is to use \保存-child-count and test the value of the childcount field.

\ifglshasdesc\{⟨entry label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries}
Tests if the given entry, which must be defined, has the description field set.

\ifglshasfield\{⟨field label⟩\}\{⟨entry label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries}
Tests if the given entry, which must be defined, has the given field set to a non-empty value.

\ifglshaslong\{⟨entry label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries}
Tests if the given entry, which must be defined, has the long field set.

\ifglshasparent\{⟨entry label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries}
Tests if the given entry, which must be defined, has the parent field set.

\ifglshasprefix\{⟨entry label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries-prefix}
Tests if the given entry, which must be defined, has the prefix field set to value that’s not empty.

\ifglshasprefixfirst\{⟨entry label⟩\}\{⟨true⟩\}\{⟨false⟩\} \textit{glossaries-prefix}
Tests if the given entry, which must be defined, has the prefixfirst field set to value that’s not empty.
General Command Summary

\ifglshasprefixfirstplural\langle entry label \rangle\langle \true \rangle\langle \false \rangle\glossaries-prefix
Tests if the given entry, which must be defined, has the prefixfirstplural field set to value that’s not empty.

\ifglshasprefixplural\langle entry label \rangle\langle \true \rangle\langle \false \rangle\glossaries-prefix
Tests if the given entry, which must be defined, has the prefixplural field set to value that’s not empty.

\ifglshasshort\langle entry label \rangle\langle \true \rangle\langle \false \rangle\glossaries
Tests if the given entry, which must be defined, has the short field set.

\ifglshassuppressedesc\langle entry label \rangle\langle \true \rangle\langle \false \rangle\glossaries
Tests if the given entry, which must be defined, has the description field set to \nopostdesc.

\ifglshassymbol\langle entry label \rangle\langle \true \rangle\langle \false \rangle\glossaries
Tests if the given entry, which must be defined, has the symbol field set to value that’s not empty and not \relax.

\ifglisused\langle label \rangle\langle \true \rangle\langle \false \rangle\glossaries
Does \langle true \rangle if the entry given by \langle label \rangle has been used, \langle false \rangle 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.

\ifglsxtrinsertinside\glossaries-extra v1.02+
Switch that determines whether or not inserted text (provided in the final optional argument of commands like \gls) is inside or outside of the font changing commands in the predefined abbreviation styles. The default is \false.

\ifignoredglossary\langle type \rangle\langle \true \rangle\langle \false \rangle\glossaries v4.08+
Tests if the glossary given by \langle type \rangle was defined as an ignored glossary.

\ifnum\langle number1 \rangle\langle comparison \rangle\langle number2 \rangle\TeX primitive*
Integer conditional.

\ifstrempty\langle string \rangle\langle \true \rangle\langle \false \rangle etoolbox
Tests if \langle string \rangle is empty.

\immediate\langle file operation \rangle\TeX primitive
Perform the file operation immediately instead of the usual delay.

\index\langle text \rangle\kernel command
Indexes the given term by writing the relevant information to an associated file that can then be processed by makeindex or xindy.

\indexname glossaries or language packages
Language-sensitive name used for the title of the glossary created with the index package option.

\input\langle file \rangle\kernel command*
Input the given file.
General Command Summary

\invfmt\{maths\}
Example command.

\jobname primitive
The current job name, which is usually the name of the main .tex file without the extension.

\L kernel command*
Produces the upper case L-slash character Ł.

\l kernel command*
Produces the lower case l-slash character ł.

\label\{id\}
kernel command*
Creates a label that can be referenced with \ref or \pageref.

\let\langle token1 \rangle\langle token2 \rangle TEX primitive*
Assigns \langle token1 \rangle to \langle token2 \rangle.

listbreak etoolbox
May be used within the handler macro of etoolbox's internal list loop commands to prematurely break the loop.

\listxadd\{list cs\}\{element\} etoolbox
Globally adds (expanded) \langle element \rangle to the list stored in the control sequence \langle list cs \rangle.

\loadglsentries\{(type)\}\{file\} glossaries
Locally redefines \glsdefaulttype to \langle type \rangle and inputs \langle file \rangle.

\longnewglossaryentry\{label\}\{key=value list\}\{description\} glossaries
Defines a new glossary entry and appends \leavemode\unskip\nopostdesc at the end of \langle description \rangle.

\longnewglossaryentry*\{label\}\{key=value list\}\{description\} glossaries-extra v1.12+
Defines a new glossary entry without appending any extra code to the end of \langle description \rangle.

\longprovideglossaryentry\{label\}\{key=value list\}\{description\} glossaries
Defines a new glossary entry if one doesn't already exist with the given label.

\mainmatter book-like classes
Switches to main matter.

\makefirstuc\{text\} mfirstuc*
Converts the first letter of \langle text \rangle to upper case.
General Command Summary

\makeglossaries \texttt{glossaries}
Opens associated glossary files to be processed by \texttt{makeindex} or \texttt{xindy}.

\makeLowercase{{\text}} \texttt{kernel command*}
Converts \texttt{(text)} to lower case.

\makeNoIdxGlossaries \texttt{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 \texttt{bib2gls}.

\makeTextLowercase{{\text}} \texttt{textcase*}
Converts \texttt{(text)} to lower case.

\makeTextUppercase{{\text}} \texttt{textcase*}
Converts \texttt{(text)} to upper case.

\MakeUppercase{{\text}} \texttt{kernel command*}
Converts \texttt{(text)} to upper case.

\mathcal{{\langle character\rangle}} \texttt{kernel command (maths mode)}
Renders the given (upper case) maths character in a calligraphic font.

\mathord{{\langle maths\rangle}} \texttt{\TeX{} primitive}
Assigns the character or sub-formula in the argument to class 0, ordinary.

\MFUnocap{{\langle word\rangle}} \texttt{mfirstuc v1.09+}
Identifies \texttt{(word)} as one that should not have its case-changed by \texttt{\capitalise\texttt{words}}
unless it occurs at the start.

\midrule \texttt{booktabs}
Horizontal rule for divider between header and main content of a tabular-like environment.

\mtxfmt{{\langle symbol\rangle}}
Example command.

\nary{{\langle text\rangle}}
Example command.

\newabbreviation[\langle key=value list\rangle][\langle label\rangle][\langle short\rangle][\langle long\rangle] \texttt{glossaries-extra}
Defines a new abbreviation.

\newacronym[\langle key=value list\rangle][\langle label\rangle][\langle short\rangle][\langle long\rangle] \texttt{glossaries}
Defines a new abbreviation. The \texttt{glossaries-extra} package redefines this to use
\texttt{\newabbreviation} with the \texttt{category} set to \texttt{acronym}.

\newcommand[\langle cs\rangle][\langle n\rangle][\langle def\rangle][\langle code\rangle] \texttt{kernel command*}
Defines a new command.

\newdualentry[\langle key=value list\rangle][\langle label\rangle][\langle short\rangle][\langle long\rangle][\langle description\rangle]
Example given in \texttt{glossaries user manual}.
\newentry{{\it label}}{{\it key=value list}}
   \hspace{1cm} glossaries-extra shortcuts
    Equivalent to \texttt{\newglossaryentry}.

\newglossary[\it log]{{\it type}}{{\it gls}}{{\it glo}}{{\it title}}
   \hspace{1cm} glossaries
   Defines a new glossary identified by \it type with the given title and associated file
   extensions used by makeindex or xindy.

\newglossary*{{\it type}}{{\it title}}
   \hspace{1cm} glossaries
   Defines a new glossary identified by \it type with the given title.

\newglossaryentry{{\it label}}{{\it key=value list}}
   \hspace{1cm} glossaries
   Defines a new glossary entry.

\newglossarystyle{{\it name}}{{\it definition}}
   \hspace{1cm} glossaries
   Defines a new glossary style called \it name.

\newignoredglossary*{\it type}
   \hspace{1cm} glossaries-v4.08+
   Defines a new ignored glossary (with hyperlinks suppressed) identified by \it type that’s
   not included in the list used by commands, such as \texttt{\printunsrtglossaries}, that
   iterate over defined glossaries.

\newnum*{\it label}{\it key=value list}
   \hspace{1cm} glossaries-extra shortcuts
   Equivalent to \texttt{\glsxtrnewnumber}.

\newrobustcmd*{\it cs}{\it n}{\it def}{\it code}
   \hspace{1cm} etoolbox
   Behaves like \texttt{\newcommand} but the newly defined command will be robust.

\newsym*{\it label}{\it key=value list}{\it symbol}
   \hspace{1cm} glossaries-extra shortcuts
   Equivalent to \texttt{\glsxtrnewsymbol}.

\newterm*{\it key=value list}{\it label}
   \hspace{1cm} glossaries’s index package option
   Defines a new glossary entry where the \it description field defaults to empty.

\hfil 559
\texttt{\number\langle value\rangle} \quad \textit{\LaTeX} \text{ primitive}\textsuperscript{*}

Expands the given numerical \langle value\rangle to a base 10 integer number stripping any leading zeros (use \texttt{\char"\langle\texttt{hex}\rangle} if the value is hexadecimal).

\texttt{\numspacefmt\{\langle symbol\rangle\}}

Example command.

\texttt{\textbackslash O}

\texttt{\textbackslash O} \quad \textit{kernel command}\textsuperscript{*}

Produces the upper case O-slash character Ø.

\texttt{\textbackslash o}

\textit{kernel command}\textsuperscript{*}

Produces the lower case o-slash character ø.

\texttt{\textbackslash OE}

\textit{kernel command}\textsuperscript{*}

Produces the upper case Ė-ligature.

\texttt{\textbackslash oe}

\textit{kernel command}\textsuperscript{*}

Produces the lower case œ-ligature.

\texttt{\oldacronym\{\langle label\rangle\}\{\langle short\rangle\}\{\langle long\rangle\}\{\langle options\rangle\}}

\textit{glossaries}

Emulates the way the old glossary package defined acronyms.

\texttt{\omicron}

\textit{glossaries-extra-bib2gls}\textsuperscript{*}

Greek letter omicron o.

\texttt{\PackageError\{\langle name\rangle\}\{\langle code\rangle\}\{\langle help\rangle\}}

\textit{kernel command}\textsuperscript{*}

Generates an error message for the package identified by \langle name\rangle.

\texttt{\pagelistname}

\textit{glossaries}

Language-sensitive name used for the location list header for some glossary styles.

\texttt{\pageref\{\langle id\rangle\}}

\textit{kernel command}\textsuperscript{*}

Cross-reference the page where \texttt{\label\{\langle id\rangle\}} occurred.

\texttt{\par}

\textit{kernel command}\textsuperscript{*}

Paragraph break.

\texttt{\parenswap\{\langle text1\rangle\}\{\langle text2\rangle\}}

Example command.

\texttt{\PGLS\{\langle options\rangle\}\{\langle label\rangle\}\{\langle insert\rangle\}}

\textit{glossaries-prefix}

Does \langle prefix\rangle\texttt{\GLS\{\langle options\rangle\}\{\langle label\rangle\}\{\langle insert\rangle\}}, where the \langle prefix\rangle is obtained from the appropriate prefix field with the text converted to upper case.

\texttt{\Pgls\{\langle options\rangle\}\{\langle label\rangle\}\{\langle insert\rangle\}}

\textit{glossaries-prefix}

Does \langle prefix\rangle\texttt{\gls\{\langle options\rangle\}\{\langle label\rangle\}\{\langle insert\rangle\}}, where the \langle prefix\rangle is obtained from the appropriate prefix field with the first letter converted to upper case.
General Command Summary

\pgls[(options)]{(label)\langle insert\rangle}
glossaries-prefix

Does \langle prefix\rangle \gls[(options)]{(label)\langle insert\rangle}, where the \langle prefix\rangle is obtained from the appropriate prefix field.

\PGLSpl[(options)]{(label)\langle insert\rangle}
glossaries-prefix

Does \langle prefix\rangle \GLSpl[(options)]{(label)\langle insert\rangle}, where the \langle prefix\rangle is obtained from the appropriate prefix field with the text converted to upper case.

\Pglspl[(options)]{(label)\langle insert\rangle}
glossaries-prefix

Does \langle prefix\rangle \glsp[(options)]{(label)\langle insert\rangle}, where the \langle prefix\rangle is obtained from the appropriate prefix field with the first letter converted to upper case.

\pglspl[(options)]{(label)\langle insert\rangle}
glossaries-prefix

Does \langle prefix\rangle \glspl[(options)]{(label)\langle insert\rangle}, where the \langle prefix\rangle is obtained from the appropriate prefix field.

\pi
Greek letter pi \(\pi\).

\printglossaries
Iterates over all non-ignored defined glossaries and performs \printglossary for each one.

\printglossary[(options)]

Inputs file created by makeindex or xindy.

\printnoidxglossaries
Iterates over all non-ignored defined glossaries and performs \printnoidxglossary for each one.

\printnoidxglossary[(options)]

Uses \TeX{} to sort, collate and list the glossary.

\printunsrtglossaries
Iterates over all non-ignored defined glossaries and performs \printunsrtglossary for each one.

\printunsrtglossary[(options)]

Display a glossary by iterating over all entries associated with that glossary in the order in which they were defined (which, with \bib2gls, should correspond to the order obtained from the sort settings given in the resource set options).

Options:

\texttt{entrycounter}={\langle boolean\rangle}
Locally enable or disable top-level enumeration (overrides \texttt{entrycounter} package option)

\texttt{groups}={\langle boolean\rangle}
Controls whether or not \printunsrtglossary (or \printunsrtinnerglossary) should insert letter group markup. There’s no visible difference for glossary styles that...
General Command Summary

don’t support letter groups (and nogroupskip is in effect) or if there’s no group information (for example, --no-group has been used)

\texttt{\textbf{\textbf{label=\{\langle label\rangle\}}}}

Creates a label for this glossary by locally using \texttt{\glsxtrsetglossarylabel\{\langle label\rangle\}}

\texttt{\textbf{\textbf{leveloffset=\{\langle n\rangle\}}}}

Makes the glossary style act as though each entry’s hierarchical level is \texttt{\langle offset\rangle} more than it actually is where \texttt{\langle offset\rangle} is either \texttt{\langle n\rangle} or is locally incremented by \texttt{\langle n\rangle} if \texttt{\langle n\rangle} starts with ++

\texttt{\textbf{\textbf{nogroupskip=\{\langle boolean\rangle\}}}}

Locally change whether or not to separate groups with a vertical space if the glossary style that support this option (overrides nogroupskip package option)

\texttt{\textbf{\textbf{nonumberlist=\{\langle boolean\rangle\}}}}

Locally change whether or not to display the location lists (overrides nonumberlist package option)

\texttt{\textbf{\textbf{nopostdot=\{\langle boolean\rangle\}}}}

Locally omit the post-description punctuation (overrides nopostdot and related package options)

\texttt{\textbf{\textbf{numberedsection=\{\langle value\rangle\}}}}

Locally change whether or not to use a numbered sectioning command (overrides numberedsection package option)

\texttt{\textbf{\textbf{prefix=\{\langle label\rangle\}}}}

Locally redefine \texttt{\glolinkprefix} for the item hypertargets and for any entry reference or cross-reference hyperlinks

\texttt{\textbf{\textbf{style=\{\langle style-name\rangle\}}}}

Use the glossary style identified by \texttt{\langle style-name\rangle} (overrides current style setting)

\texttt{\textbf{\textbf{subentrycounter=\{\langle boolean\rangle\}}}}

Locally enable or disable level 1 enumeration (overrides subentrycounter package option)

\texttt{\textbf{\textbf{target=\{\langle boolean\rangle\}}}}

Locally enables or disables the hypertargets for each item

\texttt{\textbf{\textbf{targetnameprefix=\{\langle label\rangle\}}}}

Locally assign a prefix for the hypertargets assigned to each item (if \texttt{\textbf{target=\{true\}}} to avoid duplicate target names

\texttt{\textbf{\textbf{title=\{\langle text\rangle\}}}}

Locally sets the title for this glossary

\texttt{\textbf{\textbf{tocitle=\{\langle text\rangle\}}}}

Locally sets the TOC title for this glossary

\texttt{\textbf{\textbf{type=\{\langle glossary-label\rangle\}}}}

Identifies the glossary list (\texttt{\glsdefaulttype}, if omitted)
General Command Summary

\printunsrtglossary*{⟨options⟩}{⟨code⟩}
glossaries-extra v1.12+
As \printunsrtglossary but performs ⟨code⟩ first (scoped to localise any assignments within ⟨code⟩).

\printunsrtglossaryentryprocesshook{⟨label⟩}
glossaries-extra v1.21+
Performed at each iteration of the internal loop used by \printunsrtglossary.

\printunsrtglossaryhandler{⟨label⟩}
glossaries-extra v1.12+
Used by \printunsrtglossary and \printunssrtinnerglossary to handle each entry within the loop. By default this simply does \glsxtrunsrtdo.

\printunsrtglossarypredoglossary
glossaries-extra v1.21+
Hook performed by \printunsrtglossary.

\printunsrtglossaryskipentry
glossaries-extra v1.21+
Only allowed within \printunsrtglossaryentryprocesshook this command indicates that the current entry should be skipped.

\printunsrtinnerglossary{⟨options⟩}{⟨pre code⟩}{⟨post code⟩}
glossaries-extra v1.44+
Similar to \printunsrtglossary but only contains the code that would typically be placed inside the theglossary environment. This command should either be placed inside the printunsglossarywrap environment or inside the handler macro used by \printunsrtglossary. This command is unsuitable for certain glossary styles, particularly tabular-like styles.

\ProcessOptions kernel command*
Processes supplied options.

\protect⟨token⟩
kernel command*
Protects ⟨token⟩ from expansion.

\providecommand{⟨cs⟩}{⟨n⟩}{⟨def⟩}{⟨code⟩}
kernel command*
Defines a command if it’s not already defined.

\provideglossaryentry{⟨label⟩}{⟨key=value list⟩}
glossaries
Defines a new glossary entry if one doesn’t already exist with the given label.

\provideignoredglossary{⟨type⟩}
glossaries-extra v1.12+
As \newignoredglossary but does nothing if a glossary identified by ⟨type⟩ already exists.

\provideignoredglossary*{⟨type⟩}
glossaries-extra v1.12+
As \provideignoredglossary but doesn’t suppress hyperlinks.

\ProvidesPackage{⟨name⟩}{⟨version⟩}
kernal command*
Identifies a package.

R
\ref{⟨id⟩}
kernal command*
Cross-reference the location where \label{⟨id⟩} occurred.
\refstepcounter{counter name} \hfill kernel command*

Increments the given counter in a manner compatible with the \label cross-referencing mechanism.

\renewcommand{⟨cs⟩}{⟨n⟩}{⟨def⟩}{⟨code⟩} \hfill kernel command*

Redefines an existing command.

\RequirePackage{⟨options⟩}{⟨name⟩}{⟨min version⟩} \hfill kernel command*

Loads the package identified by ⟨name⟩ from within another package.

\rgls{⟨options⟩}{⟨label⟩}{⟨insert⟩} \hfill glossaries-extra v1.21+

Like \gls but checks for the record count trigger setting (the formatting is governed by \rglsformat).

\rglsformat{⟨label⟩}{⟨insert⟩} \hfill glossaries-extra v1.21+

Used by \rgls if the record count switch is triggered.

S

\section{⟨toc title⟩}{⟨title⟩} \hfill most classes that have a concept of document sections

Section heading.

\section*{⟨title⟩} \hfill most classes that have a concept of document sections

Unnumbered section heading.

\seealsoname \hfill glossaries-extra

Language sensitive "see also" text (as from v1.42 this will be defined to \alsoname if that command exists).

\selectlanguage{⟨language name⟩} \hfill babel and polyglossia

Switch to the rules of the given language.

\setabbreviationstyle{⟨category⟩}{⟨style-name⟩} \hfill glossaries-extra

Sets the abbreviation style to ⟨style-name⟩ for the given ⟨category⟩, must be used before the abbreviation is defined.

\setcardfmt{⟨maths⟩} \hfill Example command.

\setcontentsfmt{⟨contents⟩} \hfill Example command.

\setentrycounter{⟨prefix⟩}{⟨counter⟩} \hfill glossaries

Sets up the entry’s associated counter and prefix required by \glshypernumber.

\setfmt{⟨symbol⟩} \hfill Example command.

\setglossarypreamble{⟨type⟩}{⟨code⟩} \hfill glossaries

Sets ⟨code⟩ as the preamble for the given glossary (or the default of ⟨type⟩ is omitted).

\setglossarystyle{⟨name⟩} \hfill glossaries

Sets the glossary style identified by ⟨name⟩.
General Command Summary

\setmainlanguage{⟨options⟩}{⟨language name⟩} \textit{polyglossia}
Load the main document language.

\setmembershipfmt{⟨variable(s)⟩}{⟨condition⟩}
Example command.

\setmembershiponeargfmt{⟨variable(s)⟩}{⟨condition⟩}
Example command.

\setotherlanguage{⟨options⟩}{⟨language name⟩} \textit{polyglossia}
Load a secondary document language.

\setupglossaries{⟨key⟩=⟨value list⟩} \textit{glossaries}
Applies the base glossaries options that are allowed to be changed after the package has loaded.

\showglocounter{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the counter field in the transcript.

\showglodesc{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the description field in the transcript.

\showglodescplural{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the descriptionplural field in the transcript.

\showglofield{⟨entry label⟩}{⟨internal field⟩} \textit{glossaries}
Interrupts the document build and shows the value of the given field in the transcript.

\showglofirst{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the first field in the transcript.

\showglofirstpl{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the firstplural field in the transcript.

\showgloflag{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the first use flag in the transcript.

\showgloglossaries \textit{glossaries}
Interrupts the document build and shows the list of all non-ignored glossary types in the transcript.

\showglolevel{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the entry’s hierarchical level in the transcript.

\showgloloclist{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the loclist field in the transcript.

\showglolong{⟨label⟩} \textit{glossaries}
Interrupts the document build and shows the value of the long field in the transcript.
\showglo\glosname\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{name} field in the transcript.

\showglo\gloparent\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{parent} field in the transcript.

\showglo\gloplural\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{plural} field in the transcript.

\showglo\gloshort\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{short} field in the transcript.

\showglo\glosort\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{sort} field in the transcript.

\showglo\glosarycounter\{\langle\text{type}\rangle\}\glossaries
Interrupts the document build and shows the default counter for the given glossary in the transcript.

\showglo\glosaryentries\{\langle\text{type}\rangle\}\glossaries
Interrupts the document build and shows the list of entry labels for the given glossary in the transcript.

\showglo\glosarytitle\{\langle\text{type}\rangle\}\glossaries
Interrupts the document build and shows the title of the given glossary in the transcript.

\showglo\glosymbol\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{symbol} field in the transcript.

\showglo\glosymbolplural\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{symbolplural} field in the transcript.

\showglo\glostext\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{text} field in the transcript.

\showglo\gloctype\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{type} field in the transcript.

\showglo\glouseri\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{user1} field in the transcript.

\showglo\glouserii\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{user2} field in the transcript.

\showglo\glouseriii\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{user3} field in the transcript.

\showglo\glouseriv\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{user4} field in the transcript.

\showglo\gloserv\{\langle\text{label}\rangle\}\glossaries
Interrupts the document build and shows the value of the \text{user5} field in the transcript.
General Command Summary

\showglosservi{\langle label \rangle}
   glossaries
   Intermits the document build and shows the value of the user field in the transcript.

\sii{\langle unit \rangle}
   siunitx
   Displays the unit with intelligent formatting.

\sigma
   kernel command* (maths mode)
   Greek letter sigma $\sigma$.

\sortart{\langle article \rangle}{\langle text \rangle}
   Example command.

\sortmediacreator{\langle first name(s) \rangle}{\langle surname \rangle}
   Example command.

\sortname{\langle first name(s) \rangle}{\langle surname \rangle}
   Example command.

\sortop{\langle text1 \rangle}{\langle text2 \rangle}
   Example command.

\sortvonname{\langle first name(s) \rangle}{\langle von \rangle}{\langle surname \rangle}
   Example command.

\space
   kernel command*
   Produces a space.

\SS
   kernel command*
   Produces the upper case eszett $\SS$.

\ss
   kernel command*
   Produces the lower case eszett $\ss$.

\string{\langle token \rangle}
   \TeX primitive*
   If \langle token \rangle is a control sequence it expands to the escape character followed by the control sequence name.

\strong{\langle text \rangle}
   Example command.

\subglossentry{\langle level \rangle}{\langle label \rangle}{\langle location list \rangle}
   glossaries v3.08a+
   Used in the glossary to display a sub-entry.

\surd
   kernel command* (maths mode)
   Surd symbol $\sqrt{}$.

\symbol{\langle number \rangle}
   kernel command*
   Accesses the character identified by \langle number \rangle (use \symbol\{\langle \hex \rangle\} if the number is hexadecimal).

T

\tableofcontents
   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.

567
\tabularnewline
\[ \langle \text{len} \rangle \]
kernel command

Tabular version of \ (avoids conflict with forced line breaks in paragraph column formats).

\textorpdfstring{\langle \text{TEX code} \rangle}{\langle \text{PDF text} \rangle}
hyperref

Does \langle PDF text \rangle if used in a PDF bookmark, otherwise does \langle \text{TEX code} \rangle.

\textbf{\langle text \rangle}
kernel command

Displays the given text in bold.

\textcolor{\langle model \rangle}{\langle spec \rangle}{\langle text \rangle}
color

Typesets \langle text \rangle in the given colour.

\text{\langle language \rangle}{\langle options \rangle}{\langle text \rangle}
polyglossia

Typeset \langle text \rangle according to \langle language \rangle.

\textsc{\langle text \rangle}
kernel command

Applies small-caps font to \langle text \rangle.

\textsf{\langle text \rangle}
kernel command

Displays the given text in sans-serif.

\textsmaller{\langle text \rangle}
relsize

Typesets \langle text \rangle in a font size that’s smaller than the surrounding text.

\textstyle
kernel command (maths mode)

Switch to in-line maths style (vertically compact).

\textsubscript{\langle text \rangle}
kernell command* as from 2015/01/01

Displays \langle text \rangle as a subscript.

\textsuperscript{\langle text \rangle}
kernell command*

Displays \langle text \rangle as a superscript.

\texttt{\langle text \rangle}
kernel command

Displays the given text in monospaced font.

\textweathersymbol{\langle number \rangle}
ifsym

Displays weather symbol identified by \langle number \rangle.

\TH
kernel command*

Produces the upper case thorn Þ.

\th
kernel command*

Produces the lower case thorn þ.

\the{\langle register \rangle}
\TeX{} primitive*

Expands \langle register \rangle to the current value of the register.

\theglossaryentry
glossaries

Textual representation of the glossaryentry counter, which is defined with the \texttt{entrycounter} option.
General Command Summary

\texttt{\theHentrycounter} \hspace{1cm} \textbf{glossaries}
When indexing, this is set to the \texttt{\the\langle counter\rangle} command corresponding to the current indexing counter (or, if undefined, \texttt{\the\langle counter\rangle}).

\texttt{\theHglossaryentry} \hspace{1cm} \textbf{glossaries}
Hypertarget associated with the glossaryentry counter, which is defined with the \texttt{entrycounter} option.

\texttt{\toprule} \hspace{1cm} \textbf{booktabs}
Horizontal rule for the top of a tabular-like environment.

\texttt{\TrackLangLastTrackedDialect\langle dialect\rangle} \hspace{1cm} \textbf{tracklang}
Set by commands like \texttt{\TrackLocale}.

\texttt{\TrackLocale\langle language tag\rangle}} \hspace{1cm} \textbf{tracklang} v1.3+
Tracks the given language tag.

\texttt{\transposefmt\langle maths\rangle}} \hspace{1cm} \text{Example command.}

\texttt{U}

\texttt{\u\langle character\rangle}} \hspace{1cm} \textbf{kernel command*}
Puts a breve accent over \langle character\rangle.

\texttt{\u\langle hex\rangle}} \hspace{1cm} \textbf{bib2gls}
Identifies a character by its hexadecimal code in the values of some (but not all) resource options.

\texttt{\undef\langle cs\rangle}} \hspace{1cm} \textbf{etoolbox*}
Undefines the control sequence \langle cs\rangle.

\texttt{\underline\langle text\rangle}} \hspace{1cm} \textbf{kernel command}
Underlines the given text.

\texttt{\unexpanded\langle general text\rangle}} \hspace{1cm} \texttt{\LaTeX} primitive*
Expands to the argument.

\texttt{\usepackage\langle options\rangle\langle name\rangle\langle min version\rangle}} \hspace{1cm} \textbf{kernel command*}
Loads the package identified by \langle name\rangle.

\texttt{V}

\texttt{\vec\langle character\rangle}} \hspace{1cm} \textbf{kernel command* (maths mode)}
Puts right arrow accent over \langle character\rangle.

\texttt{\vecfmt\langle symbol\rangle}} \hspace{1cm} \text{Example command.}
General Command Summary

\verb!\vert!\hfill kernel command* (maths mode)
Vertical bar delimiter |.

\verb!\write18!\hfill kernel command
\textit{⟨system call⟩}
Perform shell escape if permitted.

\verb!\xglsaccsupp!\hfill glossaries-accsupp
\textit{⟨accessible text⟩}⟨text⟩
Used by the accessibility support to interface with the accsupp package, where ⟨text⟩ is fully expanded.

\verb!\xGlsXtrSetField!\hfill glossaries-extra v1.12+
\textit{⟨entry label⟩}⟨field label⟩⟨value⟩
Globally assigns the (protected) full expansion of the given ⟨value⟩ to the field identified by ⟨field label⟩ for the entry identified by ⟨entry label⟩.

\verb!\xifinlist!\hfill etoolbox
\textit{⟨element⟩}⟨list cs⟩⟨true⟩⟨false⟩
Tests if the expansion of ⟨element⟩ is in the list stored in the control sequence ⟨list cs⟩.

\verb!\xmakefirstuc!\hfill mfirstuc* v1.01+
\textit{⟨text⟩}
Applies \texttt{\xmakefirstuc} with one level expansion of the first token of ⟨text⟩.
Bibliography


Bibliography


## Index

### Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>47, 55</td>
</tr>
<tr>
<td>&quot;</td>
<td>165, 232, 289</td>
</tr>
<tr>
<td>#</td>
<td>245, 246, 308</td>
</tr>
<tr>
<td>#</td>
<td>231, 241</td>
</tr>
<tr>
<td>#</td>
<td>55, 373</td>
</tr>
<tr>
<td>$</td>
<td>308</td>
</tr>
<tr>
<td>$</td>
<td>20, 36, 56, 133, 166, 171, 361</td>
</tr>
<tr>
<td>%</td>
<td>20, 105, 231, 425</td>
</tr>
<tr>
<td>%</td>
<td>245, 246</td>
</tr>
<tr>
<td>&amp;</td>
<td>231</td>
</tr>
<tr>
<td>&amp;</td>
<td>245, 246, 308</td>
</tr>
<tr>
<td>'</td>
<td>see apostrophe (')</td>
</tr>
<tr>
<td>(</td>
<td>182, 283, 284</td>
</tr>
<tr>
<td>{</td>
<td>20, 56, 95, 133, 166, 231, 430</td>
</tr>
<tr>
<td>}</td>
<td>20, 56, 95, 133, 166, 231, 430</td>
</tr>
<tr>
<td>}</td>
<td>245, 246</td>
</tr>
<tr>
<td>}</td>
<td>245, 246</td>
</tr>
<tr>
<td>)</td>
<td>182, 283, 284</td>
</tr>
<tr>
<td>*</td>
<td>115</td>
</tr>
<tr>
<td>&quot; (regular expression, zero or more)</td>
<td>115</td>
</tr>
<tr>
<td>&quot; (regular expression, match any)</td>
<td>115, 169</td>
</tr>
<tr>
<td>@ (command line options bib2gls)</td>
<td>47, 151</td>
</tr>
<tr>
<td>\</td>
<td>197, 286, 505</td>
</tr>
<tr>
<td>@currentHref</td>
<td>32, 505</td>
</tr>
<tr>
<td>@currentlabelname</td>
<td>32, 505</td>
</tr>
<tr>
<td>@firstofone</td>
<td>297, 506</td>
</tr>
<tr>
<td>@for</td>
<td>306, 506</td>
</tr>
<tr>
<td>@gls@hypergroup</td>
<td>40</td>
</tr>
<tr>
<td>@gobble</td>
<td>298, 506</td>
</tr>
<tr>
<td>@istfilename</td>
<td>17, 506</td>
</tr>
<tr>
<td>^</td>
<td>20, 56, 133, 166, 238, 239, 430</td>
</tr>
<tr>
<td>\</td>
<td>95, 231</td>
</tr>
<tr>
<td>^ (superscript)</td>
<td>20, 34</td>
</tr>
<tr>
<td>( switch )</td>
<td>245, 246, 308</td>
</tr>
<tr>
<td>( literal )</td>
<td>20, 34, 231</td>
</tr>
<tr>
<td>( end range )</td>
<td>166–169, 177, 282, 336, 477</td>
</tr>
</tbody>
</table>

### Abbreviation styles

<table>
<thead>
<tr>
<th>Style</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>long-em-short-em</td>
<td>65</td>
</tr>
<tr>
<td>long-hyphen-postshort-hyphen</td>
<td>546</td>
</tr>
<tr>
<td>long-noshort</td>
<td>547</td>
</tr>
<tr>
<td>long-noshort-desc</td>
<td>64</td>
</tr>
<tr>
<td>long-only-short-only</td>
<td>401, 403, 482</td>
</tr>
<tr>
<td>long-only-short-only-desc</td>
<td>548</td>
</tr>
<tr>
<td>long-postshort-user-desc</td>
<td>90</td>
</tr>
<tr>
<td>long-short</td>
<td>174</td>
</tr>
</tbody>
</table>
Index

long-short-desc .................................. 88, 89, 450, 464, 481
long-short-sc ................................... 64, 174, 179
long-short-sm .................................... 65
long-short-user ................................... 87, 458
long-short-user-desc ............................... 89
short-hyphen-long-hyphen ......................... 551
short-long ........................................ 148
short-long-desc .................................. 551
short-long-user-desc ............................... 551
short-nolong ...................................... 551
short-nolong-desc ................................ 551
\abbreviationname ................................ 506
\abbrvpluralsuffix ................................. 179, 180, 506
\see also \glsxtrabbrvpluralsuffix
\ac ................................................... 37, 506
\acronymname ...................................... 507
\acronymtype ....................................... 270, 275, 311, 507
\acrpluralsuffix .................................. 179, 507
\AE .................................................. 171, 176, 507
\ae .................................................... 171, 175, 507
\alph ............................................... 193, 507
\Alpha .............................................. 232, 507
\alpha .............................................. 314, 315, 507
animals.bib ........................................ 380, 387, 390, 462
apostrophe (') .................................... 180
applications ........................................
  arara ............................................. 17
  bibtex ........................................... 1
  convertgls2bib ................................ 310, 313–320
  kpsewhich ....................................... 16, 19, 29, 111, 310
  makeglossaries ................................ 1, 17
  makeglossaries-lite .............................. 17
  makeindex ...................................... 15, 16, 29, 212, 238
  xindy ............................................ 4, 29, 212, 238
\approx .............................................. 230, 507
\appto .............................................. 241, 507
\apptoglossarypreamble ........................... 241, 507
\see also \glossarypreamble
ASCII .............................................. 241
\AtEndDocument .................................. 1, 507
attributes .......................................... 1, 507
\see category attributes
\autoref ........................................... 172, 507, \see also \label

B
\backmatter ........................................ 182, 507
bacteria.bib ....................................... 331, 401, 462
baseunits.bib ..................................... 333, 335, 405, 407, 411, 462, 465
\bfseries ........................................... 413, 507
bib2gls-en.xml ................................... 17, 154, 283–286, 291
bib2gls.bat ........................................ 18
bib2gls.jar ......................................... 18, 27
bib2gls.sh .......................................... 18
\bibglsaliassep .................................... 282
\bibglsampersandchar ............................. 22, 308
\bibglscircumchar ................................ 22, 308
\bibglecompact .................................... 194, 283
\bibglscontributor ................................. 22, 161, 162, 306
\bibglscontributorlist ............................ 22, 161, 305
\bibglsdate ........................................ 22, 166, 307
\bibglsdategroup .................................. 298, 423
\bibglsdategrouptitle ............................ 298, 423
\bibglsdatetimeremainder ....................... 306
\bibglsdelimN ...................................... 282
\see also \bibglslastDelimN
\bibglsdollarchar .................................. 22, 163, 308
\bibglsdualprefixlabel ............................ 307
\bibgleemptygroup ................................ 296
\bibgleemptygrouptitle ............................ 296
\bibglsexternalprefixlabel ........................ 308
\bibglsfirstuc ..................................... 22, 170, 309
\bibglsflattenedchildpostsort ................. 303
\bibglsflattenedchildpresort .................... 125, 303
\bibglsflattenedchildgroup ................. 125, 301
\bibglsflattenedchildhomograph ............ 125, 301
\bibglshashchar ................................... 22, 308
\bibglshexunicodectar ............................ 165, 289
\bibglshrefchar ................................... 205, 289
\bibglshrefunicode ............................... 205, 289
\bibglshypergroup ................................ 301
\bibglshyperlink .................................. 22, 263, 264, 303
\bibglssinterloper ................................ 284
\bibglslastDelimN ................................ 282
\see also \bibglsdelimN

574
Index

\bibglslettergroup 292, 293 \bibglsnewspawnacronym 99, 279
\bibglslettergrouptitle 293, 294 \bibglsnewspawnedindexentry 100, 280
\bibglslocationgroup 286, 287 \bibglsnewspawnedindexentrysecondary
\bibglslocationgroupsep 287 \bibglsnewspawnednumber 99, 280
\bibglslocprefix 196, 197, 285, 286 \bibglsnewspawnedsymbol 99, 280
\bibglslocsuffix 286 \bibglsnewspawndualindexentry 97–100, 277
\bibglsnewcase 22, 169, 309 \bibglsnewsymbol 99, 279
\bibglsnewabbreviation 269, 270 \bibglsnewsymbol 268
\bibglsnewacronym 270 \bibglsnewtertiaryindexabbreviationentrysecondary
\bibglsnewbibtexentry 93, 276, 432 \bibglsnewtertiaryindexabbreviationentry
\bibglsnewcontributor 93, 276 \bibglsnewtertiaryindexabbreviationentrysecondary
\bibglsnewdualabbreviation 275 \bibglsnewtertiaryindexabbreviationentry
\bibglsnewdualabbreviationentry 273 \bibglsnewtertiaryindexabbreviationentrysecondary
\bibglsnewdualabbreviationentryabbreviation 273 \bibglsnewtertiaryindexabbreviationentrysecondary
\bibglsnewdualabbreviationentryabbreviationsecondary 274 \bibglsnewtertiaryindexabbreviationentrysecondary
\bibglsnewdualabbreviation 272, 465 \bibglsnumbergroup 297
\bibglsnewdualabbreviationsecondary 272 \bibglsnumbergrouptitle 297
\bibglsnewdualentry 145, 261, 270 \bibglsothergrouptitle 295
\bibglsnewdualentryabbreviation 275 \bibglsothergrouptitle 296
\bibglsnewdualentryabbreviationsecondary 270 \bibglsprimary 286
\bibglsnewdualentry 270 \bibglsprimaryprefixlabel 307
\bibglsnewdualentrysecondary 270 \bibglspagename 196, 285
\bibglsnewdualindex 271 \bibglespagesname 196, 286
\bibglsnewdualindexabbreviation 272, 465 \bibglspassim 283
\bibglsnewdualindexabbreviationsecondary 272 \bibglspassimname 283
\bibglsnewdualindex 270 \bibglspostlocprefix 197, 284
\bibglsnewdualindexentry 270 \bibglspostingnumber 283
\bibglsnewdualindexsecondary 271 \bibglsspage 281
\bibglsnewdualindexnumber 271 \bibglsssealso 281
\bibglsnewdualindexsecondary 271 \bibglsseseep 281, 282
\bibglsnewdualnumber 274 \bibglsseseep 281, 282
\bibglsnewdualsymbol 274 \bibglsssetdategrouptitle 298
\bibglsnewentry 261, 267 \bibglsssetdatetimegrouptitle 297
\bibglsnewindex 269 \bibglsssetdatevertimegrouptitle 297
\bibglsnewindexplural 269 \bibglsssetemptygrouptitle 296
\bibglsnewnumber 268 \bibglsssetlastgrouptitle 292
\bibglsnewprogenitor 96, 277 \bibglsssetlettergrouptitle 292, 293
\bibglsnewspawnabbreviation 99, 278 \bibglsssetnumbergrouptitle 296
\bibglsnewspawned 99, 278
\bibglsnewspawnedacronym 99, 279
\bibglsnewspawnedindex 97–100, 277
\bibglsnewspawnedindexplural 99, 278
\bibglsnewspawnednumber 99, 280
\bibglsnewspawnednumbersecondary 99, 280
\bibglsnewspawnedsymbol 99, 279
\bibglsnewspawnedsymbolsecondary 99, 279
\bibglsnewspawned 99, 278
\bibglsnewspawnedindex 99, 277
\bibglsnewspawnedindexplural 99, 277
\bibglsnewspawnedsymbol 99, 279
\bibglsnewspawnedsymbolsecondary 99, 279
\bibglsnewsymbol 99, 279
\bibglsnewsymbol 268
\bibglsnewsymbol 99, 279
\bibglsnewtertiaryindexabbreviationentry 275
\bibglsnewtertiaryindexabbreviationentrysecondary 276
\bibglsnewtertiaryindexabbreviationentrysecondary 276

575
Index

\bibglsetothergrouptitle 295
\bibglsettimegrouptitle 299
\bibglsetunicodegrouptitle 299
\bibglsetwidest 107, 303–305
\bibglsetwidestfallback 106, 304, 305
\bibglsetwidestfortype 107, 304, 305
\bibglsetwidestfortypefallback 107, 305
\bibglsetwidesttoplevelfallback 106, 305
\bibglsetwidesttoplevelfortypefallback 107, 305
\bibglssupplemental 205, 288
\bibglssupplementalsep 288
\bibglssupplementalsublist 205, 288
\bibglssupplementalsubsep 206, 289
\biglsterinaryprefixlabel 307
\bibglstime 22, 166, 307
\bibglstimegroup 299
\bibglstimegrouptitle 299
\bibglstitlecase 22, 170, 309
\bibglssunderscorechar 22, 308
\bibglssuseabbrvfont 78, 272
\bibglssusealias 195, 282
\bibglssuselongfont 78, 272, 276
\bibglssuse 195, 282
\bibglssusealso 195, 282
\bibglssuseabbrvfont 95, 515
\bigmathsymbols.bib 357, 437, 439
\bigoperatornamefmt 357
\binaryoperators.bib 363, 437
\boldsymbol 24, 515
\books.bib 342, 345, 416, 417, 428
boolean values

see conditionals and boolean values
\bottomrule 413, 515
\caption 32, 516
case-change 169–173, 177

see also upper case,
lower case,
title case
& sentence case
category attributes 401
aposplural 180
externallocation 203–205
glossdesc 169, 453
glossdescfont 403
glossname 23, 169, 373, 404, 452
glossnamefont 403, 404, 430, 458, 459, 464
glosssymbolfont 468
markwords 553
nohyperfirst 500
noshortplural 180
recordcount 45
targetname 128, 131
targeturl 128, 131
textformat 403, 430, 458, 464
\ce 327, 516
\chapter 516
chapter counter 188, 190
\chapter* 410, 516
\char 20, 21, 232, 315, 516
chemicalformula.bib 327, 398, 462, 465, 466
\citation 31, 92, 351, 432, 516
citations.bib 351, 432
\cite 95, 432, 434, 516
\CJK 294
\CJKenvironment 294
\cjkname 294
\CLDR (Unicode Common Locale Data Repository) 27, 221, 337, 423
\color 103, 516
command line options (bib2gls)

--break-space 30, 166, 336, 477
--cite-as-record 31, 92, 351, 432
--custom-packages 21, 35, 170, 209, 233, 237
--debug 16, 21, 28, 160, 184, 234
--dir 29, 30, 111

C
\c 63, 515
\capitalisewords 169–173, 309, 515

576
Index

--expand-fields ............... 43, 290
--force-cross-resource-refs . 14, 33
--group . 3–6, 25–27, 38, 110, 143, 151, 212, 247, 290, 291, 301, 322, 388, 400, 403, 407, 410, 414, 418, 420, 428
--help .......................... 28
--ignore-packages .............. 34, 35
--interpret ...................... 20, 30, 95
--list-known-packages .......... 21, 34
--locale ........................ 16, 29
--log-file ....................... 16, 29
--map-format ..................... 37, 184, 199
--merge-nameref-on ............. 4, 32, 33, 181, 183, 205
--merge-wrglossary-records ... 32, 200
--mfirstuc-math-protection . 13, 23, 36
--mfirstuc-protection .......... 13, 23, 35, 36
--nested-link-check ............ 13, 36, 37
--no-break-space ............... 30
--no-cite-as-record ............ 31
--no-debug ........................ 21, 28
--no-expand-fields ............. 43
--no-force-cross-resource-refs 33
--no-group ........................ 42, 143, 248
--no-interpret .................. 14, 20, 30
--no-merge-wrglossary-records . 32
--no-mfirstuc-math-protection . 36
--no-mfirstuc-protection ........ 36
--no-nested-link-check .......... 37
--no-provide-glossaries ....... 11, 44, 131, 147, 249
--no-record-count .............. 45, 46
--no-record-count-unit .......... 46
--no-support-unicode-script .... 34
--no-trim-fields ............... 43, 44
--no-verbose .................... 29
--no-warn-non-bib-fields ....... 31
--no-warn-unknown-entry-types . 31
--packages ...................... 34, 35, 170
--provides-glossaries .......... 11, 44, 390
--record-count .................. 45, 46, 210–214
--record-count-unit ............ 46
--shortcuts ...................... 37
--silent ........................ 29
--support-unicode-script ....... 33
--tex-encoding .................. 42, 105
--trim-except-fields ........... 44
--trim-fields .................... 43, 44
--trim-only-fields .............. 43, 44
--version ....................... 28
--warn-non-bib-fields .......... 8, 31, 49, 438
--warn-unknown-entry-types ..... 31

command line options (convertgls2bib)
--absorb-see .................... 313
--bibenc ........................ 310
--debug .......................... 314
--help ........................... 314
--ignore-category .............. 312
--ignore-fields ................ 312
--ignore-sort ................... 311, 312
--ignore-type ................... 311, 312
--index-conversion ............. 313
--locale ........................ 313
--no-absorb-see ................ 313
--no-ignore-category ........... 312
--no-ignore-sort ................ 311
--no-ignore-type ................ 311
--no-index-conversion .......... 313
--no-overwrite .................. 311–314
--no-preamble-only ............. 313
--no-split-on-category .......... 312
--no-split-on-type ............. 311
--overwrite ...................... 310, 314
--preamble-only ................. 313, 314
--silent ........................ 314
--space-sub ..................... 310
--split-on-category ............ 311, 312
--split-on-type ................ 311–314
--texenc ........................ 310
--verbose ........................ 314
--version ........................ 314

conditionals and boolean values 346
constants.bib .................... 324, 393
convertgls2bib.jar ............... 18
convertgls2bib.sh ............... 18

577
**Index**

cross-resource reference  ...  14, 33, 47, 102, 132, 133, 137, 138
\csuse  ...  205, 516
\currentglossary  ...  516
custom group  ...  40
date-time group  ...  39, 297, 298
date group  ...  39, 298
\DeclareOptions  ...  35, 516
\DeclareOptions*  ...  35, 516
\def  ...  306, 516
\delimN  ...  183, 282, 516
\delimR  ...  190, 194, 516
derivedunits.bib  ...  335, 405, 407, 411, 462, 465
description environment  ...  464
\descriptionname  ...  517
\detokenize  ...  112, 517
\dGls  ...  77, 81, 517
\dglsdisp  ...  172, 517,
\dglslink  ...  172, 517,
\dglspl  ...  81, 517
\DH  ...  171, 517
\dh  ...  171, 517
\diamondsuit  ...  7, 517
digraph  ...  40
\displaystyle  ...  357, 517
\DJ  ...  171, 517
\dj  ...  171, 517
document environment  ...  313
\DTLandname  ...  418, 517
\DTLformatlist  ...  161, 305, 418, 517
\DTLlistformatatlastsep  ...  517, 518
\edef  ...  457, 518
\eglupdatewidest  ...  518
\ensuremath  ...  172, 361, 364, 518
entry types
  @abbreviation  ...  64, 65, 99, 147, 178, 223–228, 269, 331, 373, 375
  @acronym  ...  65, 99, 147, 178, 223, 226, 228, 270, 318
  @bibtexentry  ...  11, 31, 52, 66, 92, 93, 223, 224, 229, 276, 342, 351, 434, 435
  @comment  ...  48
  @contributor  ...  52, 66, 92, 93, 223, 224, 276, 434
  @dualabbreviation  ...  51, 85, 86, 90, 262, 264, 275
  @dualabbreviationentry  ...  64, 68, 70, 73, 82, 83, 262–265, 273
  @dualacronym  ...  90, 275
  @dualentry  ...  47, 73, 82–85, 145, 226, 255–257, 260–265, 270, 482
  @dualentryabbreviation  ...  82, 83, 265, 273, 274
  @dualindexabbreviation  ...  68, 70, 77, 90, 263, 265, 272, 465
  @dualindexentry  ...  15, 68, 75–78, 91, 100, 262, 265, 270
  @dualindexnumber  ...  81, 263, 265, 271
  @dualindexsymbol  ...  68, 78, 81, 263, 265, 271
Index

@dualnumber 84, 274
@dualsymbol 67, 83, 84, 262, 264, 274, 411
@entry 23, 24, 60–63, 73–75, 99, 145, 223–228, 267, 313, 328, 336, 373, 380–384, 400, 407, 420, 422, 482
@index 15, 62, 63, 66, 68, 75, 92, 98–100, 122, 127, 145, 174, 178, 208, 223–228, 269, 313, 337, 375, 384–387, 420, 447, 465, 466, 482
@indexplural 63, 64, 99, 223, 226, 269, 447
@number 62, 99, 225, 228, 268, 393
@preamble 11, 14, 21, 24, 25, 55–57, 102–105, 125, 159, 208, 237, 303, 323, 325, 351, 353, 373, 375, 452
@progenitor 11, 51, 53, 95–100, 147, 277
@spawnabbreviation 99, 278, 279
@spawnacronym 99, 279
@spawndualindexentry 91, 98, 100, 280, 281
@spawnentry 99, 278
@spawnindex 98, 277
@spawnindexplural 99, 277, 278
@spawnnumber 99, 280
@spawnsymbol 99, 279, 280
@symbol 23, 61, 62, 67, 99, 208, 224, 225, 228, 268, 328, 333, 352, 357, 364, 370, 400, 407, 447, 482
@tertiaryindexabbreviationentry 73, 90, 263, 265, 275, 276
equation counter 199, 202

F

fields
access 51, 525
adoptparents 51, 95–97, 100, 147
descriptionaccess 51, 526
descriptionplural 50, 51, 54, 73, 261
descriptionpluralaccess 51, 526
duallong 51, 85–90, 177, 256, 262, 275
duallongplural 51, 85, 86
dualprefix 51, 67, 73–78, 82–85, 167, 168
dualprefixfirst 51, 67, 73–78, 82–85, 167, 168

dualprefixfirstplural 51, 67, 73–78, 82–85, 167, 168
dualprefixplural 51, 67, 73–78, 82–85, 167, 168
dualshort 51, 85, 86, 177, 256, 262, 275
dualshortplural 51, 85, 86, 180
first 36, 50–52, 149, 150, 154, 179, 337, 422, 430, 476, 483
firstaccess 52, 526
firstplural 36, 50–54, 149, 179
firstpluralaccess 52, 526
long 36, 43, 44, 50, 52, 64, 65, 77, 82–86, 90, 99, 177, 179, 226, 230, 262, 275, 331, 373, 403, 404, 450, 454, 465
longaccess 52, 526
longplural 36, 50–54, 82, 85, 179
longpluralaccess 52, 527
nonumberlist 13, 50, 184

579
Index

plural 36, 50–52, 63, 73, 78, 82, 83, 99, 149, 179, 223, 261
pluralaccess 52, 527
prefix 51, 73–78, 82–85, 167, 168
prefixfirst 51, 73–78, 82–85, 167, 168
prefixfirstplural 51, 73–78, 82–85, 167, 168
prefixplural 51, 73–78, 82–85, 167, 168
shortaccess 52, 527
shortplural 36, 50–54, 82, 85, 179, 180
shortpluralaccess 52, 528
symbolaccess 52, 528
symbolplural 50, 52, 78, 83
symbolpluralaccess 52, 528
text 36, 50–52, 63, 82, 122, 125, 149, 154, 169, 174, 178, 179, 337, 361, 364, 422, 430
textaccess 52, 528
user1 50, 54, 57, 62, 84, 85, 88, 134, 142, 148, 159, 225, 246, 395, 396, 417, 420, 423, 441, 465
user2 50, 54, 148, 396, 417–420
user3 50, 54, 148, 396, 420, 422
user4 50, 54
user5 50, 54
user6 50, 54
fields, internal
bib2gls@sort 49, 53
bib2gls@sortfallback 53
bibtextcontributor 52
bibtextentry 52, 66, 93, 435
bibtextentry@{entry-type} 52, 66, 93
bibtextentry@{entry-type} 52, 66, 93
bibtexcontributor 52
bibtexttype 52, 92
childcount 52, 117
counter 13, 52
currcount 53
currcount@{value} 53
desc 53
desclplural 54
dual 52, 257, 264
⟨field⟩endpunc 52, 154, 430, 456
firstpl 54
flag 54
group 4, 5, 8, 9, 12, 20, 31, 38, 39, 42, 52, 90, 110, 122, 126, 132, 143, 151, 208, 247, 249, 291–293, 300, 388, 405, 407, 410, 434, 438, 441, 444, 481
index 53, 199–202
indexcounter 53
level 54
location 53, 97, 181–188, 195, 438
loclist 53, 181–186
longpl 54
originalentrytype 53, 108, 152
originalid 53, 140
prevcount 54
prevcount@{value} 54
prevunitmax 54
prevunittotal 54
primarylocations 53, 185, 186, 190, 286
progenitor 53, 97
progeny 53, 96
recordcount 45, 53, 212
recordcount@{counter} 45, 53
recordcount@{counter}@{location} 46, 53
secondarygroup 53, 151, 249
secondarysort 53, 249
shortpl 54
siblingcount 53, 119
siblinglist 53, 119
textaccess 52, 49
value 53
type 54

580
Index

92, 96, 106–110, 115, 127, 132, 144–147,
154, 197, 255, 256, 266, 269, 270, 275, 291,
300, 304, 311, 312, 315, 388–391, 432, 480

file formats
  .aux 1, 17, 20, 29, 37, 40, 42, 45, 101, 112, 130,
      181–184, 199, 206, 212, 231
  .bat 18
  .bb1 351
  .bib .1, 11, 17, 20, 29, 43, 47, 103–105, 111,
      130, 138, 257, 310
  .gig 24, 29, 30
  .gls 291
  .glstex 11, 13, 17, 30, 41–45, 49, 52, 53, 56,
       69, 86, 91, 101, 105, 109–112, 130, 132, 208,
       257, 258, 267, 291, 351
  .jar 19
  .log 34, 35, 301
  .out 18
  .sh 310
  .toc
films.bib 323, 345, 416, 417
first use 530
first use flag 532
\footnote 518
\forallallglossaries 364, 518
\forallallglossaries 518
\forallallglsentries 519
  see also \forallallglossaries
& \forallallglossaries
\forallglsentries 456, 458, 519
  see also \forallallglsentries
\forlistloop 458, 519
\frontmatter 183, 519
full stop (.) 134, 135, 152, 154, 158, 197, 286,
       346, 395, 424, 432, 452–458
G
\glolinkprefix 425, 519
\glossariesextrasetup 519
glossary styles 6
altlist 395, 432, 464
altlistgroup 424
alttree 105, 106, 370, 395, 438, 446
alttreegroup 400, 444
bookindex 185, 403, 424, 425, 459, 468
index 39, 403
indexgroup 3, 38, 42, 432
indexhypergroup 40, 290, 301
list 390, 395, 413
long 390
long3col–booktabs 413
mcolalttree 437
mcolalttreegroup 400, 466
mcolindexgroup 407
super 501
topic 9, 449
tree 188, 501
treegroup 9, 388, 391
treenoname 538
glossaryentry counter 499
\glossaryheader 413, 519
\glossaryname 519
\glossarypostamble 519
  see also \apptoglossarypreamble
\glossarypreamble 241, 519
  see also \apptoglossarypreamble
glossarysubentry counter 503
\glossentry 183, 413, 519
\Glossentrydesc 164, 519
\Glossentrydesc 169, 519
\Glossentryname 519
\glossentryname 169, 404, 520
\glossentrynameother 404, 468, 469, 520
\Glossentersymbol 466–469, 520
\glossentsymbol 466–469, 520
\glossxtrsetpopts 520
\GLS 172, 520
\GlS 35, 172, 520
\gls 37, 45, 49, 67, 81, 113, 125, 130, 136, 137,
Index

thevalue ........................................... 202, 203
\glsaddall ........................................ 1, 112, 523
\glsadddallunused ............................... 182, 523
\gsaddeach ......................................... 524
\glsaddkey ........................................ 48, 116, 148, 524
\glsaddstoragekey ............................... 48, 92, 116, 454, 524
\gsautoprefix .................................. 524
\gsbackslash .................................... 22, 163, 231, 524
\gscapturedgroup ................................ 134, 524
\gsacategory ..................................... 91, 524
\gsclosebrace ................................... 22, 524
\gscurrententrylabel ......................... 422, 524
\gscurrentfieldvalue ......................... 117, 524
\gsdefaulttype ................................ 311, 315, 524
\gsdefpostdesc ................................ 62, 397, 524
\gsdefpostlink ................................ 524
\gsdesc ........................................... 484, 524
\gsdescriptionaccessdisplay .............. 525
\gsdescriptionpluralaccessdisplay ....... 525
\gsdescwidth .................................... 413, 525
\gsdisablehyper ................................ 525
\glsdisp ........................................ 172, 525, see also \glslink
\gsdoifnoexists ................................. 525
\gsdoifexists .................................. 525
\gsdoifnoexistssordo ......................... 525
\gsdisablehyper ................................ 525
\gsenabledisable ................................ 525
\gssentryaccess ................................ 525
\gsentrycounterlabel ......................... 525
\GlssEntryCounterLabelPrefix ............ 525
\gsentrydesc ................................... 169, 525
\gsentrydesc ................................... 526
\gsentrydescaccess ............................ 526
\gsentrydescplural ............................ 526
\gsentrydescplural ............................ 526
\gsentrydescpluralaccess .................... 526
\gsentryfirst .................................. 22, 526
\gsentryfirst .................................. 22, 526
\gsentryfirstaccess ......................... 22, 526
\gsentryfirstaccess ........................... 22, 526
\gsentryfirstplural ......................... 22, 526
\gsentryfirstplural ......................... 22, 526
\gsentryfirstpluralaccess .................. 526

\glsadd ........................................ 1, 80, 181–183, 188–190, 202, 203, 283, 523
\glsaddcounter ................................. 188, 523
\glsaddformat .................................. 182, 188, 203, 284
\glsaddtheHvalue .............................. 203

see also \vglssaccsupp

\glsadd ........................................ 1, 80, 181–183, 188–190, 202, 203, 283, 523
\glsaddcounter ................................. 188, 523
\glsaddformat .................................. 182, 188, 203, 284
\glsaddtheHvalue .............................. 203

582
Index

\glsentryitem 413, 526 \glsentryuseriv 22, 529
\Glsentrylong 22, 526 \Glsentryuserv 22, 529
\glsentrylong 22, 404, 526 \glsentryuserv 22, 529
\glsentrylongaccess 526 \Glsentryuservi 22, 529
\Glsentrylongpl 22, 526 \glsentryuservi 22, 529
\glsentrylongpl 22, 526 \glsexpandfields 315, 316
\glsentrylongpluralaccess 527 \glextrapostnamehook \ 188, 189, 469, 484
\Glsentryname 22, 169, 527 \glsfielddef 529
\glsentryname 16, 22, 94, 143, 157–160, 264, \glsfieldedef 529
\glsentryname . 435, 527 \glsfieldset 181, 529
\glsentryplural 22, 527 \glsfieldgdef 529
\glsentryplural 22, 527 \glsfieldxdef 529
\glsentrypluralaccess 527 \glsFindWidestLevelTwo \ 304, 529
\Glsentryprefix 527 \see also \glsFindWidestTopLevelName \ 106, 529
\glsentryprefix 527 \glsFindWidestTopLevelName 529
\glsentryprefixfixfirst 527 \Glsfirst 529
\glsentryprefixfixfirst 527 \glsfirst 530
\glsentryprefixfixfirstplural 527 \glsfirstabbrvdefaultfont 530
\glsentryprefixfixfirstplural 527 \glsfirstabbrvfont 530
\Glsentryprefix 527 \glsfirstabbrvhyphenfont 530
\glsentryprefixprefixfirst 527 \glsfirstabbrvonlyfont 530
\glsentryprefixprefixfirst 527 \glsfirstabbrvscfont 530
\Glsentryprefixfirst 527 \glsfirstabbrvemfont 530
\glsentryprefixfirst 527 \glsfirstabbrvdefaultfont 530
\glsentryprefixfirstplural 527 \glsfirstlongfontnotefont 530
\glsentryprefixfirstplural 527 \glsfirstlonghyphenfont 530
\Glsentryprefix 527 \glsfirstlongonlyfont 530
\glsentryprefixprefixfirst 527 \glsfirstlonguserfont 531
\glsentryprefixprefixfirstplural 527 \glsfirstpluralaccessdisplay 531
\Glsentryprefixfirst 527 \glsfirstlonguserfont 531
\glsentryprefixprefixfixfirstplural 527 \glsfirstpluralaccessdisplay 531
\glsentryprefixfixfirstplural 527 \glsfmtfirst 531
\Glsentryprefixfirstplural 527 \glsfmtfull 531
\glsentryprefixfirstplural 527 \glsfmtlong 531
\Glsentryprefixfirstplural 527 \glsfmtname 531
\glsentryprefixfirstpluralaccess 528 \glsfmtshort 531
\Glsentryprefixfirstpluralaccess 528 \glsfmttext 531
\Glsentryprefixfirstpluralaccess 528 \glsgroupheading 290, 291
\Glsentryprefixfirstpluralaccess 528 \glsgroupskip 531
\Glsentryprefixfirstpluralaccess 528 \glshex 135, 231, 531
\Glsentryprefixfirstpluralaccess 528

583
Index

\glshyperlink 22, 172, 264, 303, 531
\glshypernumber 200, 204, 531
\glsifcategory 404, 531
\glsignore 31, 38, 182–184, 241, 388, 452, 531
\glsinlinedescformat 532
\glsinlinedopostchild 532
\glsinlinenameformat 532
\glsinlinenodeparenchildseparater 532
\glsinlinenamepostchild 532
\glsinlineseparator 532
\glsinlinesubseparator 532
\glslabel 422, 532
\glslink 57, 58, 172, 532, see also \glsdispcounter 151, 197, 202
\glslinkformat 45, 181–184, 188, 190, 199–202, 283, 284
\glshyper 81
\glshyperoutside 532
\glslocal 532
\glsnoindex 241, 532
\glsprefix 533
\glstextformat 533
\glsthevalue 533
\glsthevalue 523
\glsval 533
\glslocalreset 533
\glslocalunset 532, 533
\glslongaccessdisplay 533
\glslongdefaultfont 533
\glslongemfont 533
\glslongextraSetwidest 106, 533
\glslongextraSetwidest see also \glslongextraupdate*widest 533
\glslongextraupdate*widest 533
\glslongfont 450, 533
\glslongfootnote*font 533
\glslonghyphenfont 533
\glslongonlyfont 401, 533
\glslongpluralaccessdisplay 534
\glslongtok 450, 534
\glslongu*erfont 265, 534
\glsname 534
\glsnameaccessdisplay 534
\glsnamefont 200, 404, 534
\glsnavhypertarget 41, 534
\glsnoexpandfields 43, 315
\glsnoindexdisplayloc 181, 182
\glsnoindexloclist 183, 534
\glsnoindexloclisthandler 183, 534
\glsnumberformat 38, 137, 183–185, 192, 193, 199, 206, 534
\glsopenbrace 22, 534
\glspatchtabularx 534
\glspercentchar 22, 535
GLSpl 535
Glspl 535
\glsquote 535
\glsrefentry 535, see also \ref
\glsrenamecommand 267, 535
\glsreset 535
\glsresetentrycounter 535
\glssee 49, 212, 313, 535
\glsseeformat 181, 282, 536
\glsseeitem see also \glsseeitem,
\glsseeitemformat
\glssee entry
& \glsseeentryformat
\glsseeentry*format 158, 536
\glsseeformat
& \glsseeentryformat
\glsseeformat
& \glsseeentryformat
\glsseeformat
& \glsseeentryformat
\glssetcategoryattribute 15, 129, 536
\glssetexpandfield 43, 315
\glssetnoexpandfield 315, 316
\glssetwidest 106, 304, 536

584
Index

\glsxtrfieldforlistloop  
\glsxtrfieldifinlist  
\glsxtrfieldlistadd & \listbreak
\glsxtrfieldforlistloop  
\glsxtrfieldifinlist  
\glsxtrfieldlistadd & \listbreak
\glsxtrfieldformatcsvlist  
\glsxtrfieldforlistloop  
\glsxtrfieldifinlist  
\glsxtrfieldlistadd & \listbreak
\glsxtrfieldformatlist  
\glsxtrfieldforlistloop  
\glsxtrfieldifinlist  
\glsxtrfieldlistadd & \listbreak
\glsxtrfieldifinlist  
\glsxtrfieldforlistloop  
\glsxtrfieldifinlist  
\glsxtrfieldlistadd & \listbreak
\glsxtrfieldformatcsvlist  
\glsxtrfieldforlistloop  
\glsxtrfieldifinlist  
\glsxtrfieldlistadd & \listbreak
\glsxtrfieldformatlist  
\glsxtrfieldforlistloop  
\glsxtrfieldifinlist  
\glsxtrfieldlistadd & \listbreak
\glsxtrfractionrules  
\glsxtrfull  
\glsxtrfullsep  
\glsxtrGeneralLatinIIrules  
\glsxtrGeneralLatinIIIrules  
\glsxtrGeneralLatinIVrules  
\glsxtrGeneralLatinVrules  
\glsxtrGeneralLatinVIrules  
\glsxtrGeneralLatinVIIrules  
\glsxtrGeneralLatinVIIIrules  
\glsxtrGeneralLatinIIrules  
\glsxtrGeneralLatinIIIrules  
\glsxtrGeneralLatinIVrules  
\glsxtrGeneralLatinVrules  
\glsxtrGeneralLatinVIrules  
\glsxtrGeneralLatinVIIrules  
\glsxtrGeneralLatinVIIIrules  
\glsxtrglossentry  
\glsxtrglossentryother  
\glsxtrgroupfield  
\GLSXTRhiername  
\GlsXtrhiername  
\GlsXtrhiernamesep  
\glsxtrhiername  
\glsxtrhyphenrules  
\glsxtrhyphensuffix  
\glsxtrifcustomdiscardperiod  
\GlsXtrIfFieldCmpNum  
\GlsXtrIfFieldEqNum  
\GlsXtrIfFieldEqXpStr  
\GlsXtrIfFieldNonZero  
\GlsXtrIfFieldCmpNum  
\GlsXtrIfFieldEqNum  
\GlsXtrIfFieldEqXpStr  
\GlsXtrIfFieldNonZero  
\GlsXtrIfFieldEqNum  
\GlsXtrIfFieldEqXpStr  
\GlsXtrIfFieldNonZero  
\GlsXtrIfFieldEqNum  
\GlsXtrIfFieldEqXpStr  
\GlsXtrIfFieldNonZero  
\GlsXtrIfFieldEqNum

586
Index

\GlsXtrSetDefaultGlsOpts \ 241, 452, 550
\Glsxtrsetdefaultglossarytype \ 550
\Glsxtrsetgrouptitle \ 5, 8, 40, 143, 290–293
\Glsxtrsetpopts \ 550
\GlsxtrsetRecordCountAttribute \ 45
\Glsxtrsetwidest \ 106, 303, 304, 551
\Glsxtrstandaloneglossarytype \ 552
\Glsxtrstandaloneentryname \ 552
\Glsxtrstandaloneentryother \ 552
\Glsxtrstandaloneglossarytype \ 552
\Glsxtrstandaloneentryname \ 552
\Glsxtrstandaloneentryother \ 552
\Glsxtrstandaloneglossarytype \ 552
\Glsxtrsuphypernumber \ 203, 205, 552
\Glsxtrsuphypernumber \ 203, 205, 552
\Glsxtruserparen \ 458, 553
\Glsxtruserprefix \ 553
\Glsxtrusersuffix \ 553
\Glsxtrword \ 553
\Glsxtrwordsep \ 553

H

handlers \ 183, 186, 435, 457, see also loops
heartsuit \ 7, 553
hyperbf \ 37, 38, 80, 182, 184, 187, 188, 284, 469, 485, 553
hyperemph \ 188, 189, 553
hyperit \ 38, 553
hyperlink \ 200, 553
hyperref \ 554

I

IETF (Internet Engineering Task Force) \ 16, 29, 214, 456
ifcase \ 196, 554
ifcsdef \ 287, 554
ifcsstrequa \ 554
ifcsstring \ 554
ifdef \ 305, 554
ifdefstrequ \ 554
ifdTLlistskipempt \ 554
IfFileExists \ 457, 554
ifglossaryexists \ 11, 390, 555
ifglossaryexists* \ 11, 390, 555
ifglsentryexists \ 110, 302, 555
ifglsfieldcseq \ 555
ifglsfielddefeq \ 555
ifglsfielddefeq
Index

\texttt{\ifgls\has\children} \hspace{2cm} 555
\texttt{\ifgls\has\desc} \hspace{2cm} 459, 555
\textit{see also} \texttt{\ifgls\has\symbol}
\& \texttt{\ifgls\has\pressedesc}
\texttt{\ifgls\has\field} \hspace{2cm} 118, 155, 396, 555
\textit{see also} \texttt{\glstextr\has\field}
\& \texttt{\glstextr\if\field\Undef}

\texttt{\ifgls\has\long} \hspace{2cm} 555
\texttt{\ifgls\has\parent} \hspace{2cm} 555
\texttt{\ifgls\has\prefix} \hspace{2cm} \textit{see also} \texttt{\ifgls\has\desc},
\texttt{\glstextr\has\field}
\& \texttt{\glstextr\if\field\Undef}

\texttt{\ifgls\has\prefix\first} \hspace{2cm} 555
\textit{see also} \texttt{\ifgls\has\desc},
\texttt{\glstextr\has\field}
\& \texttt{\glstextr\if\field\Undef}

\texttt{\ifgls\has\prefix\first\plural} \hspace{2cm} 556
\textit{see also} \texttt{\ifgls\has\desc},
\texttt{\glstextr\has\field}
\& \texttt{\glstextr\if\field\Undef}

\texttt{\ifgls\has\prefix\plural} \hspace{2cm} 556
\textit{see also} \texttt{\ifgls\has\desc},
\texttt{\glstextr\has\field}
\& \texttt{\glstextr\if\field\Undef}

\texttt{\ifgls\has\short} \hspace{2cm} 456, 556
\texttt{\ifgls\has\pressedesc} \hspace{2cm} 556
\textit{see also} \texttt{\ifgls\has\desc}

\texttt{\ifgls\has\symbol} \hspace{2cm} 556
\textit{see also} \texttt{\ifgls\has\desc},
\texttt{\glstextr\has\field}
\& \texttt{\glstextr\if\field\Undef}

\texttt{\ifgls\is\used} \hspace{2cm} 556
\textit{see also} \texttt{\glstextr\if\usedOr\Undef},
\& \texttt{\glstextr\if\first\used\is\Undef}

\texttt{\ifgls\xtr\insert\inside} \hspace{2cm} 556
\texttt{\ifiglshas\ignored\glossary} \hspace{2cm} 556
\texttt{\ifnum} \hspace{2cm} 287, 556
\texttt{\ifstremp} \hspace{2cm} 162, 556
\texttt{\ignore\glossary} \hspace{2cm} 10
\texttt{\ignore\record} \hspace{2cm} 31, 45, 49, 92, 146, 181–184, 241
\texttt{\immediate} \hspace{2cm} 1, 556
\texttt{\index} \hspace{2cm} 15, 556
\texttt{\indexname} \hspace{2cm} 556
\texttt{\input} \hspace{2cm} 1, 49, 556
\textit{interpret-preamble.bib} \hspace{2cm} 322, 323, 342, 345, 366, 416, 420, 439, 476
\texttt{interpret-preamble2.bib} \hspace{2cm} 322, 323, 342, 428, 430
\texttt{\invfmt} \hspace{2cm} 365
\texttt{\iteration} \hspace{2cm} \textit{see also} \texttt{\loops}

\texttt{\jobname} \hspace{2cm} 101, 111, 132, 557
\texttt{\l} \hspace{2cm} 171, 557
\texttt{\label} \hspace{2cm} 199, 557
\textit{see also} \texttt{\ref},
\& \texttt{\pageref}

\texttt{\label\prefixes} \hspace{2cm}
\texttt{\label\prefixes\dual.} \hspace{2cm} 47, 74, 136–138, 253, 468
\texttt{\label\prefixes\ext\langle n\rangle.} \hspace{2cm} 47, 136–140, 345
\texttt{\label\prefixes\tertiary.} \hspace{2cm} 90, 136, 137, 265
\texttt{\language\resource\file\ (.xml)} \hspace{2cm}
\texttt{\grouptitle\case\langle lc\rangle} \hspace{2cm} 40, 291
\texttt{\sentence\terminators} \hspace{2cm} 154
\texttt{\tag\page} \hspace{2cm} 196, 285
\texttt{\tag\pages} \hspace{2cm} 196, 286
\texttt{\\let} \hspace{2cm} 425, 557
\texttt{\letter\group} \hspace{2cm} 3, 39, 40, 292, 293
\texttt{\link\text} \hspace{2cm} 533
\texttt{\list\break} \hspace{2cm} 557
\textit{see also} \texttt{\forlist\loop},
\texttt{\glstextr\field\forlist\loop},
\texttt{\glstextr\field\dolist\loop}
\& \texttt{\glstextr\trend\forlist\loop}

\texttt{\list\add} \hspace{2cm} 457, 557
\texttt{\loadgls\entries} \hspace{2cm} 1, 49, 315, 557
\texttt{\locale\provider} \hspace{2cm} 27, 214, 221, 336, 423
\texttt{\location\list} \hspace{2cm} 32, 113, 114, 125, 126, 181, 185, 387
\texttt{\long\new\glossary\entry} \hspace{2cm} 83, 261, 316
\texttt{\long\new\glossary\entry*} \hspace{2cm} 268, 557
\texttt{\long\provide\glossary\entry} \hspace{2cm} 317
\texttt{\longtable\environment} \hspace{2cm} 413
\texttt{\loops} \hspace{2cm} 14, 20, 93, 117, 181, 457, 458

\texttt{\interpretpreamble\binary} \hspace{2cm} 322, 323, 342, 345, 366, 416, 420, 439, 476

\texttt{\jobname} \hspace{2cm} 101, 111, 132, 557
\texttt{\jre\ (Java\ Runtime\ Environment)} \hspace{2cm} 27, 214, 221, 336, 337, 423
\texttt{\jvm\ (Java\ Virtual\ Machine)} \hspace{2cm} 29, 42, 215, 221, 310

\texttt{\career\name} \hspace{2cm} 15, 556
\texttt{\indexname} \hspace{2cm} 556
\texttt{\input} \hspace{2cm} 1, 49, 556

589
lower case  26, 169–171, 215, 216, 234, 243, 246, 247, 256, 300, 309
\mainmatter  183, 557
\makefirstuc  13, 35, 171–173, 264, 309, 453, 515, 557
\makelglossaries  4, 238, 476, 558
\MakeLowercase  247, 558
\makenoidxglossaries  4, 558
\MakeTextLowercase  309, 558
\MakeTextUppercase  308, 452, 558
\MakeUppercase  20, 315, 558
\markuptable  373, 450, 462
\mathcal  364, 558
\mathgreek  365, 439
\mathord  363, 437
\MFUnocap  170, 558
\midrule  413, 558
\minerals  382, 387, 462
\miscsymbols  369, 444, 447, 480, 481
\mtxfmt  365
\nary  358
\newabbreviation  50, 70, 83, 103, 269, 270, 275, 311, 312, 318
\newacronym  103, 270, 275, 311, 312, 318
\newcommand  81, 267, 314, 320, 321, 558
\newdualentry  82, 311, 312, 320, 321
\newentry  316, 559
\newglossary  10, 559
\newglossary*  10, 213, 408, 559
\newglossaryentry  43, 48, 101, 149, 184, 236, 310, 313, 316
\newglossarystyle  413, 559
\newignoreglossary  10, 559
\newignoreglossary*  10, 559
\newnum  319, 559
\newrobustcmd  294, 559
\newsym  319, 559
\newterm  311, 312, 317
\NG  171, 559
\ng  171, 559
\no-interpreter-preamble  162, 208, 322, 325, 336, 342, 345, 366, 416, 420, 428, 439, 476
\nobreakspace  30, 168, 559
\NoCaseChange  171, 172, 176, 452, 559
\non-ASCII  24, 47, 205, 242, 375, 454, 456
\non-letter group  39, 40, 295, 296, 481
\nopostdesc  153, 268, 269, 313, 317, 346, 559
\number  247, 560
\numbergroup  39, 212, 296, 297
\numbersfmt  365
\O  133, 171, 560
\o  171, 560
\OE  171, 176, 560
\oe  171, 175, 560
\oldacronym  311, 312, 318, 560
\omicron  352, 353, 560
\openin_any  16, 310
\openout_any  16, 310
package options
abbreviations  8, 70, 83, 499
accssupp  51, 499
acronym  499
acronymlists  499
acronyms  499, 507
automake  1, 499
autoseeindex  499
counter  199, 499
counterwithin  499
debug  499, 536
dodef  500
docdef  500
tentrycounter  499, 500
eaclocations  500
hyperfirst  500
index  317, 424, 463, 500
indexcounter  32, 199, 500
indexcrossrefs .................................. 500  xindy ....................................... 238, 504
indexonlyfirst .................................. 500  xindygloss .................................... 504
makeindex ....................................... 500  xindynoglsnumbers ............................ 504
nogroupskip .................................... 3, 39, 500 \PackageError .................................. 35, 560
nohypertypes .................................... 500 packages
nolangwarn ...................................... 500  accsupp ........................................ 51
nolist .......................................... 500  amsmath ........................................ 34
nolong .......................................... 501  amssymb ........................................ 34
nomain .......................................... 146, 410, 432, 463, 501
nomissingglistext .............................. 501
nonumberlist ................................... 184, 398, 501
norelease ....................................... 152, 317, 501
noretdefwarn ................................... 501
nostyles ....................................... 395, 400, 403, 407, 413, 432, 463, 501
nouser .......................................... 501
notranslate ..................................... 501
totre ............................................ 501	nowarn .......................................... 501	numberedsection ................................ 501, 502	numberline ...................................... 501
numbers ......................................... 319, 501
order ............................................ 502
postdot ......................................... 152, 346, 395, 424, 432, 502
postpunc ........................................ 501, 502
record .......................................... 4, 10, 17, 32, 38, 101, 109, 112, 181, 190, 202, 212, 232, 267, 395, 424, 502
glossaries ...................................... 1, 41, 53, 54, 101, 290
glossaries–accsupp ............................. 51, 52
glossaries–extra ................................ 1, 34, 53, 54, 101, 179, 199, 202
glossaries–extra–stylemods ................... 106
glossaries–prefix ............................... 51, 67, 142, 167
glossary–bookindex ............................. 403
glossary–hypernav .............................. 41
glossary–list .................................... 424
glossary–long ................................... 413
glossary–longbooktabs ......................... 413
glossary–longextra ............................. 105, 106
glossary–mcols ................................ 407
glossary–super ................................ 501
glossary–topic ................................ 449
glossary–tree .................................. 106
graphics ........................................ 34
graphicx ........................................ 34
hyperref ........................................ 32, 34, 41, 58, 128, 205, 301
ifsym ............................................ 369, 444, 447
ifthen ........................................... 34
inputenc ........................................ 42, 47, 290

Index
Index

jmlrutils ........................................... 34 \printglossary ................................. 4, 561
lipsum .............................................. 34 \printnodidxglossaries .................. 561
longtable ........................................ 413 \printnodidxglossary ...................... 561
marvosym ...................................... 369, 444, 447 \printunsrtglossaries ................. 4, 5, 561
mfstuc ........................................... 35, 36, 171 \printunsrtglossary .......... 4, 38, 68, 103, 110,
mfstuc-english ................................. 34, 170 184, 249, 250, 291, 550, 561
mhchem ........................................... 34 entrycounter .............................. 561
MnSymbol ....................................... 34 groups ........................................ 391
natbib ............................................ 34 label ......................................... 550
pifont .......................................... 34 leveloffset .................................. 391
polyglossia .................................... 456, 457 nogroupskip .............................. 77, 562
probsoln ........................................ 34 nonumberlist .............................. 425, 562
shortvrb ........................................ 34 nopostdot .................................. 562
siunitx .......................................... 21, 34, 333, 413 numberedsection .................... 562
stix .............................................. 34, 357, 437 prefix ........................................ 562
tabularx ......................................... 534 style .......................................... 425, 562
textcase ....................................... 34, 171 subentrycounter ......................... 562
textcomp ....................................... 34 target ......................................... 110, 250, 425
tipa .............................................. 34 targetnameprefix ...................... 562
tracklang ....................................... 214, 215 title ........................................ 5, 10, 250, 425
tupgreek ........................................ 34, 325 toctitle ..................................... 562
wasysym ......................................... 34 type ............................................. 10, 425
xkeyval ......................................... 43 xspace .......................................... 34
page counter .................................... 190, 199 \printunsrtglossaryentryprocesshook
\pagelistname ..................................... 196, 560 213, 563
\pageref ....................................... 172, 200, 557, 560 \printunsrtglossaryhandler .......... 563
\par .............................................. 435, 560 \printunsrtglossarypredoglossary 563
\parenswap ..................................... 103, 104 \printunsrtglossaryskipentry .... 213, 563
people.bib ...................................... 336, 420, 422, 428, 476 printunsrtglossarywrap environment 563
period ........................................... see full stop (.) \printnoidxglossary ............ 10, 390, 563
\PGLS ........................................... 560 \ProcessOptions ............................. 35, 563
\Pgl ........................................... 560 progenitor ..................................... 95–100, 146, 147, 280, 281
\pcls ........................................... 167, 561 progeny ....................................... 95–100, 147
\PGLSPl ........................................ 561 \protect ......................................... 231, 563
\Pglsp ........................................... 561 \providecommand ......................... 21, 55, 233, 267, 321, 563
\pjlsp ........................................... 561 \provideglossaryentry .......... 316
\pi ............................................ 62, 79, 163, 561 \provideignoredglossary .......... 44, 563
post-description hook ............... 86, 89, 152, 269, 346, \provideignoredglossary* .... 11, 44, 131, 147, 249, 563
396, 418, 422, 424, 434, 435, 452 \ProvidesPackage ....................... 35, 563
post-link hook ......................... 154, 422, 430, 484
post-name hook ....................... 404, 417, 422, 424, 459, 484
primary ....................................... 66 \ref ............................................. 172, 557, 563, see also \label
\printglossaries .......................... 4, 238, 561
R

592
Index

\refstepcounter .................................. 32, 199, 564
regular expressions .................................. 115, 134, 169, 191, 230
\renewcommand .................................. 21, 55, 267, 320, 564
\RequirePackage ..................................

resource options
abbreviation-name-fallback .. 64, 77, 78, 147, 228, 272, 467
abbreviation-sort-fallback .. 64, 77, 82, 86, 89, 223–228, 404, 450, 465, 467
action .................................. 109, 110, 130, 151, 248, 424, 464
adopted-parent-field .................. 95, 97, 100, 147
alias .................................. 12, 151
alias-loc .................................. 195
append-prefix-field .................. 167, 168
append-prefix-field-cs .............. 168
append-prefix-field-cs-exceptions .. 168
append-prefix-field-exceptions .. 168
append-prefix-field-nbsp-match .. 168
bibtext-contributor-fields ........... 12, 161, 162, 305, 306, 323, 337, 342
bibtextentry-sort-fallback .......... 92, 223, 224, 229
break-at ................................ 155, 158, 161, 214, 215, 230, 233–235, 251, 259
break-marker ................................ 234, 235, 251, 259
category .................................. 13, 62, 75, 92, 98, 110, 131, 144–146, 207, 256, 266–269, 416, 422,
253, 255
compact-ranges .................. 194
counter .................................. 13, 52, 151, 256
cs-label-prefix .................. 137, 138, 468
custom-sort-fallbacks .............. 63, 223–229
date-field-format .............. 166, 167, 257
date-field-locale .............. 166, 167, 258
date-fields .............. 13, 166, 167, 307
date-sort-format 166, 167, 221, 222, 245, 252, 260, 337
date-sort-locale .............. 167, 221, 222, 245, 252, 260
date-time-field-format ............ 166, 257
date-time-field-locale ............ 166, 167, 258
date-time-field-locale ............ 166, 167, 258
date-time-fields ............ 13, 166, 167, 306
decomposition ................ 242, 252, 260
dependency-fields ........... 12, 14, 49, 112, 141
description-case-change .......... 65, 178
dual-abbrv-backlink ........ 86, 264, 265
dual-abbrv-map ........ 88, 262
dual-abbrventry-backlink ........ 264, 265
dual-abbrventry-map ........ 262
dual-backlink ................ 265
dual-break-at ................ 155, 233, 259, 467
dual-break-marker ........ 235, 259
dual-category ................ 13, 74, 145, 256, 266
dual-counter ............. 13, 256
dual-date-field-format ........ 166, 167, 257
dual-date-field-locale ........ 166, 167, 258
dual-date-fields ............ 166, 167, 258
dual-date-sort-format .......... 260
dual-date-sort-locale .......... 260
dual-date-time-field-format .......... 166, 257
dual-date-time-field-locale ........ 166, 167, 258
dual-date-time-field-format .......... 166, 257
dual-decomposition .......... 242, 260
dual-entry-backlink .......... 263–265
dual-entry-map .......... 260–264
dual-entryabbrv-backlink .......... 265
dual-field ............ 52, 257, 264, 469
dual-group-formation .......... 248, 260, 300
dual-identical-sort-action .......... 235, 259
dual-indexabbrv-backlink .......... 265
dual-indexabbrv-map .......... 263
dual-indexentry-backlink .......... 75, 265
dual-indexentry-map .......... 262
dual-indexentry-map .......... 262
dual-indexsymbol-backlink .......... 265
dual-indexsymbol-map .......... 263, 466
## Index

<table>
<thead>
<tr>
<th>Term</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>dual-letter-number-punc-rule</td>
<td>243, 260</td>
</tr>
<tr>
<td>dual-letter-number-rule</td>
<td>242, 260</td>
</tr>
<tr>
<td>dual-long-case-change</td>
<td>177, 256</td>
</tr>
<tr>
<td>dual-missing-sort-fallback</td>
<td>224, 259</td>
</tr>
<tr>
<td>dual-numeric-locale</td>
<td>245, 260</td>
</tr>
<tr>
<td>dual-numeric-sort-pattern</td>
<td>245, 260</td>
</tr>
<tr>
<td>dual-prefix</td>
<td>47, 67, 74, 136–138, 253, 307, 468</td>
</tr>
<tr>
<td>dual-short-case-change</td>
<td>177, 256</td>
</tr>
<tr>
<td>dual-short-plural-suffix</td>
<td>180</td>
</tr>
<tr>
<td>dual-sort</td>
<td>67, 68, 91, 117, 155, 156, 245, 253, 258, 259</td>
</tr>
<tr>
<td>dual-sort-field</td>
<td>67, 223, 258, 259</td>
</tr>
<tr>
<td>dual-sort-number-pad</td>
<td>235, 259, 467</td>
</tr>
<tr>
<td>dual-sort-pad-minus</td>
<td>235, 259</td>
</tr>
<tr>
<td>dual-sort-pad-plus</td>
<td>235, 259</td>
</tr>
<tr>
<td>dual-sort-replace</td>
<td>231, 259</td>
</tr>
<tr>
<td>dual-sort-rule</td>
<td>258, 259</td>
</tr>
<tr>
<td>dual-sort-suffix</td>
<td>236, 259</td>
</tr>
<tr>
<td>dual-sort-suffix-marker</td>
<td>241, 260</td>
</tr>
<tr>
<td>dual-strength</td>
<td>241, 260</td>
</tr>
<tr>
<td>dual-symbol-backlink</td>
<td>264, 265</td>
</tr>
<tr>
<td>dual-symbol-map</td>
<td>262</td>
</tr>
<tr>
<td>dual-time-field-format</td>
<td>166, 167, 258</td>
</tr>
<tr>
<td>dual-trim-sort</td>
<td>230, 259</td>
</tr>
<tr>
<td>dual-type</td>
<td>13, 41, 74, 81, 83, 107, 109, 146, 147, 255, 256, 266, 285, 286, 466</td>
</tr>
<tr>
<td>duplicate-label-suffix</td>
<td>137, 480</td>
</tr>
<tr>
<td>encapsulate-fields</td>
<td>12, 162, 163</td>
</tr>
<tr>
<td>encapsulate-fields*</td>
<td>12, 163</td>
</tr>
<tr>
<td>entry-sort-fallback</td>
<td>61, 63, 223–230, 391</td>
</tr>
<tr>
<td>entry-type-aliases</td>
<td>11, 107–109, 115, 144, 146, 223, 224, 255, 256, 328, 333, 464</td>
</tr>
<tr>
<td>ext-prefixes</td>
<td>47, 136–138, 308</td>
</tr>
<tr>
<td>field-aliases</td>
<td>12, 40, 93, 95, 147, 148, 331–337, 342, 345, 420, 428, 438</td>
</tr>
<tr>
<td>field-case-change</td>
<td>12, 13, 163, 178</td>
</tr>
<tr>
<td>field-concat-sep</td>
<td>224–230</td>
</tr>
<tr>
<td>flatten</td>
<td>110, 119, 121, 126, 148, 212, 222</td>
</tr>
<tr>
<td>flatten-lonely</td>
<td>119–121, 124, 126, 301–303</td>
</tr>
<tr>
<td>flatten-lonely-rule</td>
<td>119, 122, 126</td>
</tr>
<tr>
<td>group</td>
<td>5, 12, 38–42, 49, 52, 110, 143, 247, 405, 407</td>
</tr>
<tr>
<td>group-formation</td>
<td>143, 247, 252, 260, 291, 299, 300</td>
</tr>
<tr>
<td>hex-unicode-fields</td>
<td>13, 165, 289</td>
</tr>
<tr>
<td>identical-sort-action</td>
<td>212, 235–240, 251, 259, 391, 395, 416, 444</td>
</tr>
<tr>
<td>ignore-fields</td>
<td>12, 64, 119, 126, 140, 147–152, 337</td>
</tr>
<tr>
<td>interpret-fields</td>
<td>13, 163, 165, 178</td>
</tr>
<tr>
<td>interpret-fields-action</td>
<td>165</td>
</tr>
<tr>
<td>interpret-label-fields</td>
<td>12, 20, 105, 132, 430</td>
</tr>
<tr>
<td>interpret-preamble</td>
<td>14, 24, 25, 56, 102, 105, 239, 247, 322</td>
</tr>
<tr>
<td>label-prefix</td>
<td>12, 47, 87, 131, 136–140, 248, 307, 345, 416, 468</td>
</tr>
<tr>
<td>labelify</td>
<td>12, 20, 34, 93, 105, 127, 132–135</td>
</tr>
<tr>
<td>labelify-list</td>
<td>12, 20, 34, 105, 132–135, 141</td>
</tr>
<tr>
<td>labelify-replace</td>
<td>93, 133–135, 230</td>
</tr>
<tr>
<td>letter-number-punc-rule</td>
<td>218, 243, 252, 260</td>
</tr>
<tr>
<td>letter-number-rule</td>
<td>218, 219, 242, 244, 252, 260</td>
</tr>
<tr>
<td>limit</td>
<td>13, 116, 117</td>
</tr>
<tr>
<td>loc-counters</td>
<td>197–199, 286, 287</td>
</tr>
<tr>
<td>loc-prefix</td>
<td>17, 186, 196, 197, 284–286</td>
</tr>
<tr>
<td>loc-suffix</td>
<td>197, 286, 452</td>
</tr>
<tr>
<td>long-case-change</td>
<td>177, 256</td>
</tr>
<tr>
<td>master</td>
<td>11, 13, 44, 110, 130</td>
</tr>
<tr>
<td>master-resources</td>
<td>132</td>
</tr>
<tr>
<td>match</td>
<td>13, 115, 116, 420</td>
</tr>
<tr>
<td>match-action</td>
<td>116</td>
</tr>
<tr>
<td>match-op</td>
<td>115, 116</td>
</tr>
<tr>
<td>max-loc-diff</td>
<td>193, 194, 283</td>
</tr>
<tr>
<td>min-loc-range</td>
<td>182, 190</td>
</tr>
<tr>
<td>missing-parent-category</td>
<td>127, 128</td>
</tr>
<tr>
<td>missing-parents</td>
<td>126–128</td>
</tr>
<tr>
<td>missing-sort-fallback</td>
<td>223–230, 251, 259</td>
</tr>
<tr>
<td>name-case-change</td>
<td>13, 110, 169, 174–178, 452</td>
</tr>
<tr>
<td>no-case-change-Cs</td>
<td>172, 177</td>
</tr>
</tbody>
</table>
Index

not-match .......................... 116
numeric-locale .............. 220, 245, 252, 260
numeric-sort-pattern 220, 245, 252, 260
post-description-dot ........ 152, 153
prefix-fields ................. 167, 168
primary-dual-dependency ...... 253
primary-location-formats ... 185–188
progenitor-type .............. 96, 147
progeny-type ................... 96, 147
record-label-prefix ........... 137, 473
replicate-fields .............. 12, 65, 95, 134, 147–150, 337, 403, 422, 430, 438, 465
replicate-missing-field-action 149, 150
replicate-override ............ 149, 150, 337
save-child-count ............. 117, 119
save-index-counter ............ 32, 199–202
save-locations ............... 4, 110, 181, 184, 199, 418, 441
save-loclist .................... 185, 186, 267
save-original-entrytype 12, 108, 144, 146, 151, 152
save-original-entrytype-action 144, 146, 152
save-original-id .............. 12, 140
save-primary-locations ... 185–188, 286
save-sibling-count .......... 119
secondary ...................... 11, 44, 53, 60, 66, 109, 110, 147, 151, 224, 245, 248, 249, 423, 464
secondary-break-at .......... 233, 251
secondary-break-marker ...... 235, 251
secondary-date-sort-format .... 252
secondary-date-sort-locale 245, 252
secondary-decomposition .... 242, 252
secondary-group-formation ... 248, 252, 300
secondary-identical-sort-action 235, 251
secondary-letter-number-punc-rule 243, 252
secondary-letter-number-rule 242, 252
secondary-missing-sort-fallback
secondary-numeric-locale .... 245, 252
secondary-numeric-sort-pattern 245, 252
secondary-sort-number-pad . 235, 251
secondary-sort-pad-minus ... 235, 251
secondary-sort-pad-plus .... 235, 251
secondary-sort-replace ...... 231, 251
secondary-sort-rule .......... 249, 251
secondary-sort-suffix ...... 236, 252
secondary-sort-suffix-marker 241, 252
secondary-strength .......... 241, 252
secondary-trim-sort ......... 230, 251
see .................................. 195, 281
seealso ......................... 49, 195
set-widest 20, 105–107, 137, 303–305, 358, 370, 395, 400, 438, 449, 480
short-case-change .......... 12, 65, 130, 175–177, 256
short-plural-suffix .......... 12, 180
shuffle ......................... 117, 212, 222
sort-field ................. 5, 23–25, 64, 147, 208, 209, 222–229, 248, 250, 259, 352, 395, 413, 447
sort-label-list .......... 155, 160, 207
sort-number-pad ........... 216, 230, 235, 251, 259
sort-pad-minus .......... 235, 251, 259
sort-pad-plus .............. 235, 251, 259
sort-replace ............. 135, 230, 234, 251, 259
sort-rule .......... 20, 21, 210, 214, 231, 251, 259
sort-suffix .......... 236, 237, 252, 259, 417, 438, 444
sort-suffix-marker .... 238, 241, 252, 260
src ......................... 14, 17, 102, 111, 113, 139, 202, 206
strength .................. 241, 242, 252, 260
strip-missing-parents .... 126, 127
strip-trailing-nopost .... 13, 153, 346
Index

suffixF ........................................... 194
suffixFF ......................................... 194
supplemental-category ....................... 205, 207
supplemental-locations ....................... 203, 288
supplemental-selection ....................... 206
symbol-sort-fallback 61, 208, 223–225, 228, 229, 328, 391, 395, 413, 467
tertiary-category .............................. 90, 266
tertiary-prefix ................................. 90, 137, 265, 307
tertiary-type ..................................... 90, 109, 147, 266
time-field-format .............................. 166, 167, 258
time-field-locale ............................... 166, 167, 258
time-fields ....................................... 13, 166, 167, 307
trigger-type ...................................... 11, 44, 45, 146, 147
trim-sort ........................................ 230, 251, 259
unknown-entry-alias ......................... 92, 109
word-boundaries ................................ 170, 177
write-preamble .................................. 21, 105, 209, 323, 430
\rgls ............................................. 45, 46, 146, 183
\rglsformat ...................................... 45

S

sample-authors.tex ......................... 127, 428
sample-bacteria.tex ......................... 401, 417, 464
sample-chemical.tex ......................... 398, 467
sample-citations.tex ......................... 432
sample-constants.tex ......................... 393
sample-dual.tex ................................ 320
sample-hierarchical.tex ...................... 387–391
sample-markuplanguages.tex ................ 450, 464
sample-maths.tex ............................. 366, 439
sample-media.tex .............................. 322, 346, 416, 430, 479
sample-msymbols.tex ......................... 437–441
sample-multi1.tex .............................. 15, 462, 473, 478, 483, 484
sample-multi2.tex .............................. 15, 473
sample-nested.tex .............................. 10, 113, 388
sample-people.tex ............................. 420, 430, 476, 477, 483
sample-textsymbols.tex ...................... 5, 39, 369, 438, 444, 447
sample-textsymbols2.tex ..................... 5, 39, 438, 446, 447
sample-units1.tex ............................. 405, 408
sample-units2.tex ............................. 408
sample-units3.tex ............................. 411, 467
sample-usergroups.tex ......................... 454
secondary 66, see also dual
section counter ................................. 203
\section ......................................... 32, 564
\section* ......................................... 410, 463, 564
\seealso ....................................... 49, 314, 315, 564
\selectlanguage .................................. 160, 564
sentence case .................................... 171
\setabbreviationstyle .......................... 64, 103, 564
\setcardfmt ...................................... 365
\setcontentsfmt ................................ 365
\setentrycounter ................................ 205, 564
\setfmt .......................................... 365
\setglossarypreamble .......................... 564
\setglossarystyle .............................. 413, 564
\setmainlanguage .............................. 456, 565
\setmembershipfmt ............................. 365
\setmembershiponeargfmt .................... 365
\setotherlanguage .............................. 458, 565
\setupglossaries ................................ 565
\showglocounter ................................ 565
\showglodesc .................................. 565
\showglodescplural ............................. 565
\showglofield .................................. 565
\showglofirst .................................. 565
\showglofirstpl ................................ 565
\showgloflag .................................... 565
\showgloglossaries ............................ 565
\showglolevel .................................. 565
\showgloloclist ................................ 565
\showglolong ................................... 565
\showgloname .................................. 566
\showgloparent ................................ 566
\showgloplural ................................ 566

596
Index

\showgloshort .......................... 566
\showglossort .......................... 566
\showglossarycounter .................. 566
\showglossaryentries .................. 566
\showglossarytitle ..................... 566
\showglosymbol .......................... 566
\showglosymbolplural ................... 566
\showglotext ............................. 566
\showglotype ............................ 566
\showglouseriv ........................... 566
\showglouserii ........................... 566
\showglouseriii ........................... 566
\showglouserv ............................. 566
\showglouservi ............................ 566
\showglouservi ............................ 567
\si ........................................... 21, 172, 333, 413, 567
si unit ...................................... 333, 413, 479
\sigma ........................................ 164, 567
\sortart ...................................... 322, 323
\sortmediacreator ........................ 323, 430
\sortname ................................... 322, 323, 423, 483
\sortop ...................................... 208, 209, 322
\sortvonname ................................ 322, 323, 337, 423, 483
\space ....................................... 167–169, 177, 567
\SS ........................................... 171, 567
\ss ........................................... 171, 567
StackOverflowError ...................... 20
\string ....................................... 135, 168, 230, 231, 241, 245, 246
\strng ....................................... 103–105
\subglossentry ............................. 183, 414, 567
\surd ......................................... 324, 567
switches
see command line options (bib2gls)
& command line options (convertgls2bib)
\symbol ...................................... 289, 567
symbol group
see non-letter group

T

\tableofcontents .......................... 164, 567
\tabularnewline ............................ 413, 568
tabularx environment .................... 534
\terms.bib ................................. 386, 387, 390, 481–483
tertiary .................................... 90
\texorpdfstring ............................ 165, 568
texparserlib.jar ........................... 18–21, 24, 28, 34
\textbf ...................................... 181, 182, 192, 434, 568
\textcolor .................................. 176, 568
\text{(language) ............................. 458, 568
\textsc ...................................... 77, 174, 568
\textsf ...................................... 417, 568
\textsmaller ................................ 520, 568
\textstyle ................................... 357, 568
\textsuperscript ........................... 33, 244, 568
\textsuperscript ........................... 33, 244, 568
\texttt ....................................... 237, 568
\textweathersymbol ....................... 370, 568
\TH .......................................... 171, 568
\th .......................................... 171, 568
\the ........................................... 450, 568
\theglossary environment ............... 563
\theglossaryentry .......................... 568
\theHentrycounter .......................... 32, 569
\theHglossaryentry ......................... 569
time group ................................ 39, 299
title case ................................... 170, 171, 177, 178, 309
toc (table of contents) .................. 17
\topics.bib ................................... 387, 447
\toprule ....................................... 413, 569
\TrackLocale ................................ 457, 569
\TrackLangLastTrackedDialect ............ 457, 569
\transposesymf ............................. 365
trigraph ..................................... 40
TUG (\TeX Users Group) ................... 375

U

\u (accent) .................................. 231, 569
\u (hex code) .............................. 168, 229–231, 241, 245, 246
\unaryoperators.bib ....................... 364, 437
\undef ....................................... 444, 569
\underline ................................... 452, 569
\unexpanded ................................ 293, 294, 569

Unicode categories
Letter, Lowercase ......................... 218
Letter, Modifier ............................ 218
Letter, Other ............................... 218

597
Index

Letter, Titlecase ................. 218
Letter, Uppercase ................ 218
Number, Decimal Digit .......... 191, 216, 244
Punctuation, Close ............. 153, 154
Punctuation, Dash ............... 177
Punctuation, Final quote ........ 153, 154
Punctuation, Other .............. 153, 154
Separator, Space ................ 218

upper case . 23, 26, 36, 40, 65, 170, 171, 176, 215,
216, 234, 243, 308, 309, 452, 453
\usepackage ...................... 51, 314, 501, 569
usergroups.bib .................. 375, 454

\vec .......................... 24, 569
\vecfmt ......................... 365

vegetables.bib .................. 384, 387, 462
\vert .......................... 25, 57, 365, 570

wrglossary counter .......... 32, 53, 199, 200
\write18 ......................... 1, 570

X

xampl.bib ....................... 93
\xglsaccsupp .................... 570
see also \glsaccsupp
\xGlsXtrSetField ............... 457, 570
\xifinlist ...................... 457, 570
\xmakefirstuc ................... 143, 570
XML resource file
see language resource file (.xml)