

fixtounicode

Clea F. Rees*

v0.1.1 (SVN Rev: 11683) 2026/02/24

Abstract

fixtounicode provides interfaces for adding ‘tounicode’ mappings to type1 and/or T_EX fonts which lack them. expl3 and L^AT_EX 2_ε interfaces are provided, the former being designed primarily to support type1 symbol packages.

Contents

1	Usage	1
2	Programming interface	2
3	Implementation	3

1 Usage

`fixtounicode` (*pkg.*) `\usepackage` [*package options*] {`fixtounicode`}

loads the package with *package options*.

Available options:

`debug` (*opt.*) = `true` | `false`

Load debugging code, which generates more verbose console and log messages.

`dingbats` (*opt.*) = `true` | `false`

Enable support for Zapf Dingbats if compiling with luaT_EX. This option is not really specific to Dingbats, but that is probably the best known case. Enabling this option when compiling with luaT_EX 1.24 or later will load the Unicode code points known to pdfT_EX (unless DVI output is enabled).

The option does nothing if a different or older engine is used. For pdfT_EX, it is unnecessary; the mappings are enabled by default. For other engines and older luaT_EX, the mappings would either be silently ignored (luaT_EX) or raise errors (e.g. luaTeX in DVI mode).

`\fixtounicode` {*comma-separated list of key-values*}

Add tounicode mappings for T_EX/type1 fonts which lack them.

Supported keys:

default Default code point.

glyphs Comma-separated list of glyph names. For use with unicodes.

*Bug tracker: codeberg.org/cfr/nfssex/issues | Code: codeberg.org/cfr/nfssex | Mirror: github.com/cfr42/nfssex

mappings Key-value list of glyph to Unicode mappings. pdfTeX and LuaTeX 1.24 and later. Probably unsuitable for package code.

pfb Type1 font.

tfm TFM.

unicodes Comma-separated list of code points. For use with `glyphs`.

For example, the following adds mappings for three of `marvosym`'s symbols.

```
\usepackage{fixtounicode}
\fixtounicode{%
  tfm = umvs,
  mappings = {%
    Coffeecup = 26BD,
    Radioactivity = 2615,
    Football = 2622,
  },
}
\usepackage{marvosym}
```

When the corresponding symbols are used in the document,

```
\Coffeecup\quad
\Radioactivity\quad
\Football
```

the result is a PDF with the specified Unicode code points. This means that copy-paste works correctly, provided the viewing application supports Unicode, screen-reading software has the relevant semantic data and conversions to text etc. work as expected.

Packages which need to define sets of mappings should use either `glyphs` and `unicodes` or the `expl3` functions in section 2. `mappings` is intended for specific corrections or additions at the document level.

2 Programming interface

```
\fixtounicode_tounicode:nnNN{<TFM>}{<PFB>}{<sequence 1>}{<sequence 2>}
```

```
\fixtounicode_tounicode:nNN {<font name>}{<sequence 1>}{<sequence 2>}
```

where `<TFM>` and `<PFB>` are the names of the TFM and PFB without extensions, `<sequence 1>` is the name of a sequence variable containing glyph names and `<sequence 2>` is the name of a sequence variable containing the corresponding Unicode code points. `\fixtounicode_tounicode:nnNN` is a convenience wrapper for the common case in which `<TFM>` and `<PFB>` are the same. In that case `` is their common name, without extension.

```
\fixtounicode_tounicode:nnnn{<TFM>}{<PFB>}{<sequence 1>}{<sequence 2>}
```

```
\fixtounicode_tounicode:nnn {<font name>}{<sequence 1>}{<sequence 2>}
```

Wrappers around `\fixtounicode_tounicode:nnNN` and `\fixtounicode_tounicode:nNN` which accept is comma-separated lists of glyph names and code points in place of variables. These versions simply set sequence variables from the comma-separated lists and pass their names to the underlying functions.

3 Implementation

You do not need to read the remainder of this document in order to install or use the package.

<*sty> <@@=fixtounicode>

```

1 \GetIdInfo $Id: fixtounicode.dtx 11683 2026-02-24 03:57:46Z cfrees $
2 {Fix missing tounicode values in TeX fonts}
3 \ProvidesExplPackage{\ExplFileName}
4 {\ExplFileDate}{v0.1.1 \ExplFileVersion}{\ExplFileDescription}

```

debug (*key*) Keys.

dingbats (*key*)

```

5 \keys_define:nn { fixtounicode }
6 {
7   debug .bool_set:N = \l__fixtounicode_debug_bool,
8   debug .initial:n = false,
9   debug .default:n = true,
10  dingbats .bool_set:N = \l__fixtounicode_dingbats_bool,
11  dingbats .default:n = true,
12  dingbats .initial:n = false,
13 }
14 \ProcessKeyOptions[fixtounicode]

```

code_glyphtounicode_seq (*var.*) Variables.

code_tounicode_seq (*var.*)

code_glyphs_seq (*var.*)

code_unicode_seq (*var.*)

code_engine_seq (*var.*)

code luatex_has_fix_bool (*var.*)

```

15 \seq_new:N \l__fixtounicode_glyphtounicode_seq
16 \seq_new:N \l__fixtounicode_tounicode_seq
17 \seq_new:N \l__fixtounicode_glyphs_seq
18 \seq_new:N \l__fixtounicode_unicodes_seq
19 \seq_new:N \g__fixtounicode_engine_seq
20 \bool_new:N \g__fixtounicode_luatex_has_fix_bool

```

Format variant.

```

21 \cs_generate_variant:Nn \seq_set_item:Nnn { NnV }

```

__fixtounicode_debug:n (*fn.*)

__fixtounicode_debug:N (*fn.*)

```

22 \bool_if:NTF \l__fixtounicode_debug_bool
23 {
24   \cs_new_protected:Npn \__fixtounicode_debug:n #1
25   {
26     \iow_term:n {
27       [fixtounicode debug]:: #1
28     }
29   }
30   \cs_new_protected:Npn \__fixtounicode_debug:N #1
31   {
32     \__fixtounicode_debug_aux:eV { \cs_to_str:N #1 } #1
33   }
34   \cs_new_protected:Npn \__fixtounicode_debug_aux:nn #1#2
35   {
36     \__fixtounicode_debug:n { #1 ->#2 }
37   }
38   \cs_generate_variant:Nn \__fixtounicode_debug_aux:nn {eV}
39   \sys_if_engine luatex:T
40   {
41     \lua_now:n { fixtounicode_debug = true }
42   }

```

```

43 }{
44  \cs_new_eq:NN \__fixtounicode_debug:n \use_none:n
45  \cs_new_eq:NN \__fixtounicode_debug:N \use_none:n
46 }

```

If the LuaTeX version is less than 1.24, set dev to false; o/w true.

```

47 \sys_if_engine luatex:T
48 {
49  \__fixtounicode_debug:n {Engine is LuaTeX.}
50  \bool_set_true:N \l__fixtounicode luatex_has_fix_bool
51  \seq_gset_split:NnV \g__fixtounicode_engine_seq {.} \c_sys_engine_version_str
52  \int_compare:nNnTF { \seq_item:Nn \g__fixtounicode_engine_seq {1} } < {1}
53  {
54    \__fixtounicode_debug:n {
55      LuaTeX version 0 or less:
56      you get out too much.
57      Stay home and update!
58    }
59    \bool_set_false:N \l__fixtounicode luatex_has_fix_bool
60  } {
61    \int_compare:nNnT { \seq_item:Nn \g__fixtounicode_engine_seq {2} } < {24}
62    {
63      \__fixtounicode_debug:n {
64        LuaTeX minor version less than 24.
65        I'll do my best, but binary too old for full support.
66      }
67      \bool_set_false:N \l__fixtounicode luatex_has_fix_bool
68    }
69  }
70  \__fixtounicode_debug:N \l__fixtounicode luatex_has_fix_bool
71 }

```

This is not good at all

```

72 \sys_ensure_backend:

```

`\pdfglyphtounicode` LuaTeX only, from the manual.

```

73 \bool_lazy_and:nnT
74 { \sys_if_engine luatex_p: }
75 { \sys_if_output_pdf_p: }
76 {
77  \protected\def\pdfglyphtounicode {\pdfextension glyphtounicode }
78  \lua_now:n { pdf.setgentounicode(1) }
79  \bool_if:NT \l__fixtounicode luatex_has_fix_bool { \input glyphtounicode.tex\relax }
80  \bool_if:NT \l__fixtounicode dingbats_bool
81  {
82    \input glyphtounicode.tex \relax
83  }
84 }

85 \msg_new:nnn { fixtounicode } { mapping }
86 {
87  \msg_info_text:n { fixtounicode } ::
88  Mapping #1 ->#2 \msg_line_context:
89 }
90 \msg_new:nnn { fixtounicode } { limitations }
91 {
92  \msg_warning_text:n { fixtounicode } ::
93  Sorry, use of #1 is not supported on #2 \msg_line_context:.
94  #3

```

```

95 }
96 \msg_new:nnn { fixtounicode } { file-awol }
97 {
98   \msg_error_text:n { fixtounicode } ::
99   #1 not found on \msg_line_context:
100 }

```

`\fixtounicode_tounicode:nnNN` (*fn.*) Engine-specific functions for mappings. If we're using pdfTeX, things are straightforward: we simply use the `tfm:` syntax for the primitive `\pdfglyphtounicode`. If we're using LuaTeX, things are complicated. For newer releases (2026), the macro `\pdfglyphtounicode` defined above will accept `tfm:`. Otherwise, we could apply values directly to the `tfm`, but that will only work for the newer releases (which don't need it), so we apply them to the type1 `pfb` for now, using a workaround from Max Chernoff.

```

101 \cs_new_protected_nopar:Npn \__fixtounicode_tounicode_pair:nn #1#2 {}
102 \bool_if:nT
103 {
104   \bool_lazy_and_p:nn
105   {
106     \bool_lazy_or_p:nn {\sys_if_engine_pdftex_p:}
107     {
108       \bool_lazy_and_p:nn
109       {\sys_if_engine luatex_p:}
110       {\l__fixtounicode luatex_has_fix_bool}
111     }
112   } {\sys_if_output_pdf_p:}
113 } {
114   \__fixtounicode_debug:n {
115     TFM specific tounicode mappings supported.
116     Enabling support for glyph names.
117   }
118   \cs_new_protected_nopar:Npn \__fixtounicode_tounicode:nnn #1#2#3
119   {
120     \cs_set_nopar:Npn \__fixtounicode_tounicode_pair:nn ##1##2
121     {

```

TFM-specific mapping.

pdfTeX manual page 33.

```

122   \pdfglyphtounicode { tfm:#1/##1 } { ##2 }
123   \msg_info:nnnn {fixtounicode} {mapping} {tfm:#1/##1} {##2}
124   }
125   \seq_set_eq:NN \l__fixtounicode_glyph_tounicode_seq #2
126   \seq_set_eq:NN \l__fixtounicode_tounicode_seq #3
127   \seq_map_pairwise_function:NNN \l__fixtounicode_glyph_tounicode_seq
128   \l__fixtounicode_tounicode_seq \__fixtounicode_tounicode_pair:nn
129   }
130   \cs_new_protected_nopar:Npn \__fixtounicode_tounicode:nnNN #1#2#3#4
131   {
132     \__fixtounicode_tounicode:nnn { #1 } #3 #4
133   }
134 }

```

For LuaTeX without the `dev` option, we use the workaround mentioned above. This deals with a limitation in the engine (lack of support for the `tfm:` syntax in `\pdfextension tounicode`) and a bug (failure to recognise any tounicode mappings specified for `tfms`).

```

135 \bool_lazy_all:nT
136 {
137   {\sys_if_engine luatex_p:}
138   {!\l__fixtounicode luatex_has_fix_bool}

```

```

139 {\sys_if_output_pdf_p:}
140 } {
141   \__fixtounicode_debug:n {
142     TFM specific tounicode mappings unsupported: LuaHBTeX too old.
143     Enabling workaround.
144     Note glyph names unsupported.
145     Support requires indices.
146   }
147   \lua_load_module:n { fixtounicode }
148   \cs_set_nopar:Npn \__fixtounicode_tounicode_pair:nn #1#2
149   {
150     \lua_now:n {
151       table.insert(fixtounicodeTab,tonumber("#2",16))
152     }
153   }
154   \cs_new_protected_nopar:Npn \__fixtounicode_tounicode:nnNN #1#2#3#4
155   {
156     \__fixtounicode_debug:n {
157       Trying to add tounicode mappings for old LuaTeX.
158       Fingers crossed!
159     }
160     \__fixtounicode_debug:n {TFM: #1; PFB: #2;}
161     \__fixtounicode_debug:N #3
162     \__fixtounicode_debug:N #4
163     \lua_now:n { fixtounicodeTab = {} }
164     \seq_set_eq:NN \l__fixtounicode_glyphtounicode_seq #3
165     \seq_set_eq:NN \l__fixtounicode_tounicode_seq #4
166     \seq_map_pairwise_function:NNN \l__fixtounicode_glyphtounicode_seq
167     \l__fixtounicode_tounicode_seq \__fixtounicode_tounicode_pair:nn
168     \lua_now:e {
169       fixtounicode.tounicodes("#1","#2",fixtounicodeTab)
170     }
171   }
172   \cs_new_protected_nopar:Npn \__fixtounicode_tounicode:nnN #1#2#3
173   {
174     \__fixtounicode_tounicode:nnNN { #1 } { #1 } #2 #3
175   }
176 }

```

On all other engines, the functions are noop.

```

177 \sys_if_output_dvi:T
178 {
179   \__fixtounicode_debug:n {
180     tounicode mappings unsupported for DVI.
181     Installing noop functions.
182   }
183   \cs_new_eq:NN \__fixtounicode_tounicode:nnNN \use_none:nnnn
184   \cs_new_eq:NN \__fixtounicode_tounicode:nNN \use_none:nnn
185 }
186 \cs_generate_variant:Nn \__fixtounicode_tounicode:nnNN { VVNN }

```

`__fixtounicode_tounicode:nnNN` (*fn.*) Public expl3.

`__fixtounicode_tounicode:nNN` (*fn.*)

`__fixtounicode_tounicode:nnnn` (*fn.*) 187 \cs_new_eq:NN \fixtounicode_tounicode:nnNN __fixtounicode_tounicode:nnNN

`__fixtounicode_tounicode:nnn` (*fn.*) 188 \cs_new_eq:NN \fixtounicode_tounicode:nNN __fixtounicode_tounicode:nnN

`__fixtounicode_tounicode:nn` (*fn.*) 189 \cs_new_protected_nopar:Npn \fixtounicode_tounicode:nnnn #1#2#3#4

```

190 {
191   \seq_set_split:Nnn \l__fixtounicode_glyphs_seq { , } { #3 }
192   \seq_set_split:Nnn \l__fixtounicode_unicodes_seq { , } { #4 }
193   \__fixtounicode_tounicode:nnNN { #1 } { #2 }
194   \l__fixtounicode_glyphs_seq \l__fixtounicode_unicodes_seq

```

```

195 }
196 \cs_new_protected_nopar:Npn \fixtounicode_tounicode:nnn #1#2#3
197 {
198   \seq_set_split:Nnn \l__fixtounicode_glyphs_seq { , } { #2 }
199   \seq_set_split:Nnn \l__fixtounicode_unicodes_seq { , } { #3 }
200   \__fixtounicode_tounicode:nnNN { #1 } { #1 }
201   \l__fixtounicode_glyphs_seq \l__fixtounicode_unicodes_seq
202 }

default (key) Keys.
glyphs (key)
mappings (key) 203 \keys_define:nn { fixtounicode }
                204 {
pfb (key)       205   default .tl_set:N = \l__fixtounicode_default_tl,
tfm (key)       206   default .initial:n = 2FFFF,
unicodes (key)  207   default .default:V = \c_empty_tl,
                208   glyphs .clist_set:N = \l__fixtounicode_glyphs_clist,
                209   glyphs .value_required:n = true,
                210   mappings .code:n = {
                211     \prop_put_from_keyval:Nn \l__fixtounicode_mappings_prop { #1 }
                212   },

                213   pfb .tl_set:N = \l__fixtounicode_pfb_tl,
                214   pfb .initial:V = \c_empty_tl,
                215   tfm .tl_set:N = \l__fixtounicode_tfm_tl,
                216   tfm .initial:V = \c_empty_tl,
                217   unicodes .clist_set:N = \l__fixtounicode_unicodes_clist,
                218   unicodes .value_required:n = true,
                219 }

                220 \prop_new:N \l__fixtounicode_mappings_prop

```

`__fixtounicode_tounicode:` (*fn.*) Generic interface.

```

221 \cs_new_protected:Npn \__fixtounicode_tounicode:
222 {
223   \tl_if_empty:NT \l__fixtounicode_tfm_tl
224   {
225     \tl_if_empty:NTF \l__fixtounicode_pfb_tl
226     {
227       \msg_error:nnnV {fixtounicode} {file-awol} \l__fixtounicode_pfb_tl
228     }{
229       \tl_set_eq:NN \l__fixtounicode_tfm_tl \l__fixtounicode_pfb_tl
230     }
231   }
232   \tl_if_empty:NT \l__fixtounicode_pfb_tl
233   {
234     \tl_set_eq:NN \l__fixtounicode_pfb_tl \l__fixtounicode_tfm_tl
235   }
236   \seq_set_from_clist:NN \l__fixtounicode_glyphs_seq \l__fixtounicode_glyphs_clist
237   \seq_set_from_clist:NN \l__fixtounicode_unicodes_seq \l__fixtounicode_unicodes_clist
238   \seq_map_indexed_inline:Nn \l__fixtounicode_unicodes_seq
239   {
240     \tl_if_eq:VnTF \c_empty_tl { ##2 }
241     {
242       \seq_set_item:NnV \l__fixtounicode_unicodes_seq { ##1 } \l__fixtounicode_default_tl
243     } {
244       \tl_if_eq:nnT { ##2 } { 0 }
245       {
246         \seq_set_item:NnV \l__fixtounicode_unicodes_seq { ##1 } \l__fixtounicode_default_tl
247       }

```

```

248   }
249 }

```

We now use any key-value mappings set *via* the `mappings` key. This is intended for cases where only a few mappings are needed from a particular font. Font support packages providing mappings for symbol fonts should preferably use the public `expl3` functions or, failing that, the keys `glyphs` and `unicodes` as I expect the implementation of `mappings` to be significantly slower.

Note that this method is **NOT** currently supported on Lua \TeX . To test on Lua \TeX , install a development binary, (re)generate appropriate formats and load this package with the `dev` option. Note the binary should probably **not** be used for real documents.

For \TeX Live, the 2026 pretest contains a version of Lua \TeX which supports this functionality, so it is no longer necessary to download a separate binary or regenerate formats.

On pdf \TeX this method should work just fine.

```

250 \prop_if_empty:NF \l__fixtounicode_mappings_prop
251 {
252   \bool_lazy_and:nnTF
253   { \sys_if_engine luatex_p: }
254   { ! \l__fixtounicode luatex_has_fix_bool }
255   {
256     \msg_warning:nnnnn { fixtounicode } { limitations }
257     { key mappings }
258     { this version of LuaTeX }
259     { Please update your TeX distribution.
260       If you cannot update,use the keys glyphs and unicodes
261         or equivalent expl3 functions.
262       Alternatively,compile with pdfTeX.
263     }
264   }{
265     \prop_map_inline:Nn \l__fixtounicode_mappings_prop
266     {
267       \seq_put_right:Nn \l__fixtounicode_glyphs_seq { ##1 }
268       \seq_put_right:Nn \l__fixtounicode_unicodes_seq { ##2 }
269     }
270   }
271 }

272 \__fixtounicode_tounicode:VVNN \l__fixtounicode_tfm_tl \l__fixtounicode_pfb_tl
273   \l__fixtounicode_glyphs_seq \l__fixtounicode_unicodes_seq
274 }

```

`\fixtounicode` 2e syntax.

```

275 \NewDocumentCommand \fixtounicode { +m }
276 {
277   \group_begin:
278     \keys_set:nn { fixtounicode } { #1 }
279     \__fixtounicode_tounicode:
280   \group_end:
281 }

```

</sty>

Change History

v0.1.1

General: First public release. 1

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
<code>__fixtounicode_debug:N</code> (fn.)	22 , 70 , 161 , 162
<code>__fixtounicode_debug:n</code> (fn.)	22 , 49 , 54 , 63 , 114 , 141 , 156 , 160 , 179
<code>__fixtounicode_debug_aux:eV</code>	32
<code>__fixtounicode_debug_aux:nn</code>	34 , 38
<code>__fixtounicode_tounicode:</code> (fn.)	221 , 279
<code>__fixtounicode_tounicode:VVNN</code> (fn.)	101 , 272
<code>__fixtounicode_tounicode:nNN</code> (fn.)	101 , 188
<code>__fixtounicode_tounicode:nnNN</code> (fn.)	101 , 187 , 193 , 200
<code>__fixtounicode_tounicode_pair:nn</code> (fn.)	101
B	
<code>\bool_if:NT</code>	79 , 80
<code>\bool_if:nT</code>	102
<code>\bool_if:NTF</code>	22
<code>\bool_lazy_all:nT</code>	135
<code>\bool_lazy_and:nnT</code>	73
<code>\bool_lazy_and:nnTF</code>	252
<code>\bool_lazy_and_p:nn</code>	104 , 108
<code>\bool_lazy_or_p:nn</code>	106
<code>\bool_new:N</code>	20
<code>\bool_set_false:N</code>	59 , 67
<code>\bool_set_true:N</code>	50
C	
<code>\c_empty_tl</code>	207 , 214 , 216 , 240
<code>\c_sys_engine_version_str</code>	51
<code>\cs_generate_variant:Nn</code>	21 , 38 , 186
<code>\cs_new_eq:NN</code>	44 , 45 , 183 , 184 , 187 , 188
<code>\cs_new_protected:Npn</code>	24 , 30 , 34 , 221
<code>\cs_new_protected_nopar:Npn</code>	101 , 118 , 130 , 154 , 172 , 189 , 196
<code>\cs_set_nopar:Npn</code>	120 , 148
<code>\cs_to_str:N</code>	32
D	
<code>debug</code> (key)	5
<code>debug</code> (opt.)	1
<code>\def</code>	77
<code>default</code> (key)	203
<code>dingbats</code> (key)	5
<code>dingbats</code> (opt.)	1
E	
expl3 functions:	
<code>__fixtounicode_debug:N</code>	22 , 70 , 161 , 162
<code>__fixtounicode_debug:n</code>	22 , 49 , 54 , 63 , 114 , 141 , 156 , 160 , 179
<code>__fixtounicode_tounicode:</code>	221 , 279
<code>__fixtounicode_tounicode:VVNN</code>	101 , 272
<code>__fixtounicode_tounicode:nNN</code>	101 , 188
<code>__fixtounicode_tounicode:nnNN</code> 101 , 187 , 193 , 200	
<code>__fixtounicode_tounicode_pair:nn</code>	101
<code>\fixtounicode_tounicode:nNN</code>	187
<code>\fixtounicode_tounicode:nnn</code>	187
<code>\fixtounicode_tounicode:nnNN</code>	187
<code>\fixtounicode_tounicode:nnmn</code>	187
expl3 variable:	
<code>\l_fixtounicode_engine_seq</code>	15
<code>\l_fixtounicode_glyphs_seq</code>	15 , 191 , 194 , 198 , 201 , 236 , 267 , 273
<code>\l_fixtounicode_glyph_tounicode_seq</code>	15 , 125 , 127 , 164 , 166
<code>\l_fixtounicode luatex_has_fix_bool</code>	15 , 50 , 59 , 67 , 70 , 79 , 110 , 138 , 254
<code>\l_fixtounicode_tounicode_seq</code>	15 , 126 , 128 , 165 , 167
<code>\l_fixtounicode_unicode_seq</code>	15
<code>\ExplFileDate</code>	4
<code>\ExplFileDescription</code>	4
<code>\ExplFileName</code>	3
<code>\ExplFileVersion</code>	4
F	
<code>\fixtounicode</code>	1 , 275
<code>fixtounicode</code> (pkg.)	1
<code>\fixtounicode_tounicode:nNN</code>	2

<code>\fixtounicode_tounicode:nNN</code> (fn.)	187	<code>\l__fixtounicode_tounicode_seq</code> (var.)	15, 126, 128, 165, 167
<code>\fixtounicode_tounicode:nnn</code>	2	<code>\l__fixtounicode_unicode_seq</code> (var.)	15
<code>\fixtounicode_tounicode:nnn</code> (fn.)	187	<code>\l__fixtounicode_unicodes_clist</code>	217, 237
<code>\fixtounicode_tounicode:nnNN</code>	2	<code>\l__fixtounicode_unicodes_seq</code>	18, 192, 194, 199, 201, 237, 238, 242, 246, 268, 273
<code>\fixtounicode_tounicode:nnNN</code> (fn.)	187	<code>\lua_load_module:n</code>	147
<code>\fixtounicode_tounicode:nnnn</code>	2	<code>\lua_now:e</code>	168
<code>\fixtounicode_tounicode:nnnn</code> (fn.)	187	<code>\lua_now:n</code>	41, 78, 150, 163
G			
<code>\g__fixtounicode_engine_seq</code>	19, 51, 52, 61	M	
<code>\g__fixtounicode luatex_has_fix_bool</code>	20	mappings (key)	203
<code>\GetIdInfo</code>	1	<code>\msg_error:nnnV</code>	227
glyphs (key)	203	<code>\msg_error_text:n</code>	98
<code>\group_begin:</code>	277	<code>\msg_info:nnnn</code>	123
<code>\group_end:</code>	280	<code>\msg_info_text:n</code>	87
I			
<code>\input</code>	79, 82	<code>\msg_line_context:</code>	88, 93, 99
<code>\int_compare:nNnT</code>	61	<code>\msg_new:nnn</code>	85, 90, 96
<code>\int_compare:nNnTF</code>	52	<code>\msg_warning:nnnnn</code>	256
<code>\iow_term:n</code>	26	<code>\msg_warning_text:n</code>	92
K			
<code>\keys_define:nn</code>	5, 203	N	
<code>\keys_set:nn</code>	278	<code>\NewDocumentCommand</code>	275
L			
<code>l3keys:</code>		O	
<code>debug</code>	5	options:	
<code>default</code>	203	<code>debug</code>	1
<code>dingbats</code>	5	<code>dingbats</code>	1
<code>glyphs</code>	203	P	
<code>mappings</code>	203	packages:	
<code>pfb</code>	203	<code>fixtounicode</code>	1
<code>tfm</code>	203	<code>\pdfextension</code>	77
<code>unicodes</code>	203	<code>\pdfglyphtounicode</code>	73, 122
<code>\l__fixtounicode_debug_bool</code>	7, 22	<code>pfb</code> (key)	203
<code>\l__fixtounicode_default_tl</code>	205, 242, 246	<code>\prop_if_empty:Nf</code>	250
<code>\l__fixtounicode_dingbats_bool</code>	10, 80	<code>\prop_map_inline:Nn</code>	265
<code>\l__fixtounicode_engine_seq</code> (var.)	15	<code>\prop_new:N</code>	220
<code>\l__fixtounicode_glyphs_clist</code>	208, 236	<code>\prop_put_from_keyval:Nn</code>	211
<code>\l__fixtounicode_glyphs_seq</code> (var.)	15, 191, 194, 198, 201, 236, 267, 273	<code>\protected</code>	77
<code>\l__fixtounicode_glyphtounicode_seq</code> (var.)	15, 125, 127, 164, 166	<code>\ProvidesExplPackage</code>	3
<code>\l__fixtounicode luatex_has_fix_bool</code> (var.)	15, 50, 59, 67, 70, 79, 110, 138, 254	R	
<code>\l__fixtounicode_mappings_prop</code>	211, 220, 250, 265	<code>\relax</code>	79, 82
<code>\l__fixtounicode_pfb_tl</code>	213, 225, 227, 229, 232, 234, 272	S	
<code>\l__fixtounicode_tfm_tl</code>	215, 223, 229, 234, 272	<code>\seq_gset_split:NnV</code>	51
		<code>\seq_item:Nn</code>	52, 61
		<code>\seq_map_indexed_inline:Nn</code>	238
		<code>\seq_map_pairwise_function:NNN</code>	127, 166
		<code>\seq_new:N</code>	15, 16, 17, 18, 19
		<code>\seq_put_right:Nn</code>	267, 268
		<code>\seq_set_eq:NN</code>	125, 126, 164, 165

