

The fontspec package

Font selection for X_YLaTeX and LuaLaTeX

WILL ROBERTSON

With contributions by Khaled Hosny,
Philipp Gesang, Joseph Wright, and others.

<http://wsprr.io/fontspec/>

2024/02/13 v2.9a

Contents

I fontspec.dtx	5
1 Package declaration	5
1.1 Lua header	6
II fontspec-code-load.dtx	7
1 The fontspec.sty loading file	7
III fontspec-code-vars.dtx	8
1 Declaration of variables	8
IV fontspec-code-msg.dtx	13
1 Error/warning/info messages	13
1.1 Errors	13
1.2 Warnings	14
1.3 Info messages	16
V fontspec-code-opening.dtx	18
1 Opening code	18
1.1 Package options	18
1.2 Encodings	18

1.3	Generic functions	19
1.4	expl3 variants	20
VI	fontspec-code-fontload.dtx	21
1	expl3 interface for primitive font loading	21
VII	fontspec-code-interfaces.dtx	23
1	User commands	23
VIII	fontspec-code-user.dtx	27
1	User command internals	27
1.1	Font selection	27
1.2	Font feature selection	30
1.3	Defining new font features	32
1.4	High level conditionals	34
1.5	\oldstylenums and \liningnums	35
IX	fontspec-code-api.dtx	36
1	Programmer's interface	36
X	fontspec-code-internal.dtx	42
1	Internals	42
1.1	The main function for setting fonts	42
1.2	Setting font shapes in a family	50
1.3	Initialisation	60
1.4	Miscellaneous	61
XI	fontspec-code-opentype.dtx	63
1	OpenType definitions code	63
1.1	Adding features when loading fonts	64
1.2	OpenType feature information	68
XII	fontspec-code-graphite.dtx	72
1	Graphite/AAT code	72

XIII	fontspec-code-keyval.dtx	74
1	Font loading (keyval) definitions	74
1.1	Pre-pre-parsing stages	74
1.2	Pre-parsed features	77
1.3	Font faces	77
1.4	General font-independent features	81
XIV	fontspec-code-feat-opentype.dtx	91
1	OpenType feature definitions	91
2	Regular key=val / tag definitions	91
2.1	Ligatures	91
2.2	Letters	91
2.3	Numbers	92
2.4	Vertical position	92
2.5	Contextuals	92
2.6	Diacritics	93
2.7	Kerning	93
2.8	Fractions	93
2.9	Style	94
2.10	CJK shape	94
2.11	Character width	95
2.12	Vertical	95
3	OpenType features that need numbering	95
3.1	Alternate	95
3.2	Variant / StylisticSet	96
3.3	CharacterVariant	96
3.4	Annotation	97
3.5	Ornament	97
4	Script and Language	97
4.1	Script	97
4.2	Language	98
5	Backwards compatibility	99
XV	fontspec-code-scripts.dtx	100
1	Font script definitions	100
XVI	fontspec-code-lang.dtx	104
1	Font language definitions	104

XVII	fontspec-code-feat-aat.dtx	112
1	AAT feature definitions	112
1.1	Ligatures	112
1.2	Letters	112
1.3	Numbers	113
1.4	Contextuals	113
1.5	Diacritics	113
1.6	Vertical position	113
1.7	Fractions	113
1.8	Alternate	113
1.9	Variant / StylisticSet	114
1.10	Style	114
1.11	CJK shape	114
1.12	Character width	115
1.13	Annotation	115
XVIII	fontspec-code-enc.dtx	116
1	Extended font encodings	116
XIX	fontspec-code-math.dtx	119
1	Selecting maths fonts	119
XX	fontspec-code-closing.dtx	124
1	Closing code	124
1.1	Finishing up	124
XXI	fontspec-code-xfss.dtx	125
1	Changes to the NFSS	125
1.1	Italic small caps and so on	125
1.2	Strong emphasis	125
Index		128

File I

fontspec.dtx

1 Package declaration

List all dtx files for running the ins file and typesetting the code.

```
1 <*dtx>
2 \gdef\FONTSPECDTX{
3   \DTX{fontspec.dtx}
4   \DTX{fontspec-code-load.dtx}
5   \DTX{fontspec-code-vars.dtx}
6   \DTX{fontspec-code-msg.dtx}
7   \DTX{fontspec-code-opening.dtx}
8   \DTX{fontspec-code-fontload.dtx}
9   \DTX{fontspec-code-interfaces.dtx}
10  \DTX{fontspec-code-user.dtx}
11  \DTX{fontspec-code-api.dtx}
12  \DTX{fontspec-code-internal.dtx}
13  \DTX{fontspec-code-opentype.dtx}
14  \DTX{fontspec-code-graphite.dtx}
15  \DTX{fontspec-code-keyval.dtx}
16  \DTX{fontspec-code-feat-opentype.dtx}
17  \DTX{fontspec-code-scripts.dtx}
18  \DTX{fontspec-code-lang.dtx}
19  \DTX{fontspec-code-feat-aat.dtx}
20  \DTX{fontspec-code-enc.dtx}
21  \DTX{fontspec-code-math.dtx}
22  \DTX{fontspec-code-closing.dtx}
23  \DTX{fontspec-code-xfss.dtx}
24 }
25 </dtx>
```

Now exit if we're using plain \TeX ; this would usually be the case when loading this file with `fontspec.ins`.

```
26 <*dtx>
27 \def\tmpa{plain}
28 \ifx\tmpa\fmtname\expandafter\endinput\fi
29 </dtx>
```

Metadata for documentation; the official title and authors of the package.

```
30 <*dtx>
31 \title{
32   The \textsf{fontspec} package\\
33   Font selection for \XeLaTeX\ and \LuaLaTeX
34 }
35 \author{
36   \textsc{Will Robertson}\\
37   With contributions by Khaled Hosny,\\
38   Philipp Gesang, Joseph Wright, and others.\\
39   \url{http://wspr.io/fontspec/}
```

```

40 }
41 </dtx>

```

Declare the package version and date for each of the .sty files generated. In addition, declare the version and date for this .dtx file.

```

42 <fontspec>\RequirePackage{xparse}
43 <fontspec & load>\ProvidesExplPackage{fontspec}%
44 <fontspec & XE>\ProvidesExplPackage{fontspec-xetex}%
45 <fontspec & LU>\ProvidesExplPackage{fontspec-luatex}%
46 *dtx)
47 \RequirePackage{xparse}
48 \ProvidesExplFile{fontspec.dtx}
49 </dtx>
50 *fontspec)
51 {2024/02/13}{2.9a}{Font selection for XeLaTeX and LuaLaTeX}
52 </fontspec>

```

Here the version and date are setup for typesetting the documentation.

```

53 *dtx)
54 \GetFileInfo{fontspec.dtx}
55 \date{\filedate \quad \fileversion}
56 </dtx>

```

1.1 Lua header

```

57 <lua>fontspec          = fontspec or {}
58 <lua>local fontspec   = fontspec
59 <lua>fontspec.module  = {
60 <lua>  name           = "fontspec",
61 <lua>  version        = "2.9a",
62 <lua>  date           = "2024/02/13",
63 <lua>  description     = "Font selection for XeLaTeX and LuaLaTeX",
64 <lua>  author         = "Khaled Hosny, Philipp Gesang, Will Robertson",
65 <lua>  copyright      = "Khaled Hosny, Philipp Gesang, Will Robertson",
66 <lua>  license        = "LPPL v1.3c"
67 <lua>}

```

File II

fontspec-code-load.dtx

1 The fontspec.sty loading file

Before we begin, for the rest of the package we use the `@@ expl3` module syntax with module name 'fontspec'.

```
1 <@@=fontspec>
```

The fontspec.sty file is simply set up to load the appropriate fontspec-xetex.sty or fontspec-luatex.sty file. This is performed by the following code.

```
2 <*load>
```

Lua[®]TeX

```
3 \sys_if_engine_luatex:T
4 {
5   \RequirePackage{luaotfload}
6   \lua_now:e{require("fontspec")}
7   \RequirePackage{fontspec-luatex}
8   \endinput
9 }
```

X₃TeX

```
10 \sys_if_engine_xetex:T
11 {
12   \RequirePackage{fontspec-xetex}
13   \endinput
14 }
```

Other If not one of the above, error and exit.

```
15 \msg_new:nnn {fontspec} {cannot-use-pdftex}
16 {
17   The~ fontspec~ package~ requires~ either~ XeTeX~ or~ LuaTeX.\\
18   You~ must~ change~ your~ typesetting~ engine~ to,~ e.g.,~
19   "xelatex"~ or~ "lualatex"~ instead~ of~ "latex"~ or~ "pdflatex".
20 }
21 \msg_fatal:nn {fontspec} {cannot-use-pdftex}
```

Closing That's the end of the fontspec.sty file.

```
22 \endinput
23 </load>
```

File III

fontspec-code-vars.dtx

1 Declaration of variables

This file consists solely of declaration of variables used by fontspec. In some cases these variables are also initialised with default values. In time I would like to move these initialisations

Booleans

`\l_@@_firsttime_bool` As `\keys_set:nn` is run multiple times, some of its information storing only occurs once while we decide if the font family has been defined or not. When the later processing is occurring per-shape this no longer needs to happen; this is indicated by the ‘firsttime’ conditional.

```
1 \bool_new:N \l_@@_firsttime_bool
```

(End of definition for \l_@@_firsttime_bool. This function is documented on page ??.)

```
2 \bool_new:N \l_@@_nobf_bool
```

```
3 \bool_new:N \l_@@_noit_bool
```

```
4 \bool_new:N \l_@@_nosc_bool
```

```
5 \bool_new:N \l_@@_check_bool
```

```
6 \bool_new:N \l_@@_tfm_bool
```

```
7 \bool_new:N \l_@@_atsui_bool
```

```
8 \bool_new:N \l_@@_ot_bool
```

```
9 \bool_new:N \l_@@_mm_bool
```

```
10 \bool_new:N \l_@@_harfbuzz_bool
```

```
11 \bool_new:N \l_@@_graphite_bool
```

```
12 \bool_new:N \l_@@_fontcfg_bool
```

```
13 \bool_set_true:N \l_@@_fontcfg_bool
```

For dealing with legacy maths:

```
14 \bool_new:N \g_@@_math_euler_bool
```

```
15 \bool_new:N \g_@@_math_lucida_bool
```

```
16 \bool_new:N \g_@@_pkg_euler_loaded_bool
```

For package options:

```
17 </fontspec>
```

```
18 <*options>
```

```
19 \bool_new:N \g_@@_cfg_bool
```

```
20 \bool_new:N \g_@@_math_bool
```

```
21 </options>
```

```
22 <*fontspec>
```

```
23 \bool_new:N \l_@@_tmpa_bool
```

```
24 \bool_new:N \l_@@_disable_defaults_bool
```

```
25 \bool_new:N \l_@@_alias_bool
```

```
26 \bool_new:N \l_@@_external_bool
```

```
27 \bool_new:N \l_@@_defining_encoding_bool
```

```
28 \bool_new:N \l_@@_scriptlang_exist_bool
```

```
29 \bool_new:N \g_@@_em_normalise_slant_bool
```

```
30 \bool_new:N \l_@@_external_kpse_bool
```

```
31 \bool_new:N \l_@@_proceed_bool
```


`\l_@@_never_check_bool` It is used to disable checking opentype script, language, and tags when running checking code that has a user-defined return path we want to allow the higher-level code to dictate the logic. TODO: tidy this up!

```
32 \bool_new:N \l_@@_never_check_bool
```

(End of definition for \l_@@_never_check_bool. This function is documented on page ??.)

Counters

```
33 \int_new:N \l_@@_script_int
34 \int_new:N \l_@@_language_int
35 \int_new:N \l_@@_strnum_int
36 \int_new:N \l_@@_tmp_int
37 \int_new:N \l_@@_tmpa_int
38 \int_new:N \l_@@_tmpb_int
39 \int_new:N \l_@@_tmpc_int
40 \int_new:N \l_@@_em_int
41 \int_new:N \l_@@_emdef_int
42 \int_new:N \l_@@_strong_int
43 \int_new:N \l_@@_strongdef_int
```

Floats

```
44 \fp_new:N \l_@@_tmpa_fp
45 \fp_new:N \l_@@_tmpb_fp
```

Dimensions

```
46 \dim_new:N \l_@@_tmpa_dim
47 \dim_new:N \l_@@_tmpb_dim
48 \dim_new:N \l_@@_tmpc_dim
```

Sequences

```
49 \seq_new:N \l_@@_bf_series_seq
```

Comma-lists

```
50 \clist_new:N \g_@@_default_fontopts_clist
51 \clist_new:N \g_@@_all_keyval_modules_clist
52 \clist_new:N \l_@@_sizefeat_clist
53 \clist_set:Nn \l_@@_sizefeat_clist {Size={-}}
54 \clist_new:N \l_@@_extensions_clist
55 \clist_new:N \l_@@_fontopts_clist
56 \clist_new:N \l_@@_family_fontopts_clist
57 \clist_new:N \l_@@_all_features_clist
58 \clist_new:N \l_@@_leftover_clist
59 \clist_new:N \l_@@_keys_leftover_clist
60 \clist_new:N \l_@@_sizing_leftover_clist
61 \clist_new:N \l_@@_fontfeat_clist
62 \clist_new:N \l_@@_fontfeat_curr_clist
63 \clist_new:N \l_@@_arg_clist
64 \clist_new:N \l_@@_this_feat_clist
```

```

65 \clist_new:N \l_@@_fontfeat_up_clist
66 \clist_new:N \l_@@_fontfeat_bf_clist
67 \clist_new:N \l_@@_fontfeat_it_clist
68 \clist_new:N \l_@@_fontfeat_bfit_clist
69 \clist_new:N \l_@@_fontfeat_sl_clist
70 \clist_new:N \l_@@_fontfeat_bfsl_clist
71 \clist_new:N \l_@@_fontfeat_sw_clist
72 \clist_new:N \l_@@_fontfeat_bfsw_clist
73 \clist_new:N \l_@@_fontfeat_sc_clist

```

Property lists

```

74 \prop_new:N \g_@@_fontopts_prop
75 \prop_new:N \l_@@_nfss_prop
76 \prop_new:N \l_@@_nfssfont_prop
77 \prop_new:N \g_@@_OT_features_prop
78 \prop_new:N \g_@@_all_opentype_feature_names_prop
79 \prop_new:N \g_@@_em_prop
80 \prop_new:N \g_@@_strong_prop
81 \prop_new:N \g_@@_fontid_family_prop
82 \prop_new:N \g_@@_family_int_prop
83 \prop_new:N \g_@@_rawvariations_prop

```

Token lists

Visible (perhaps?)

```

84 \tl_new:N \l_fontspeg_family_tl
85 \tl_new:N \g_fontspeg_encoding_tl
86 \tl_new:N \l_fontspeg_fontname_tl

```

2e interactions

```

87 \tl_clear_new:N \UTFencname
88 \tl_clear_new:N \cyrillicencoding
89 \tl_clear_new:N \latinencoding

```

Renderer/shaper

```

90 \tl_new:N \l_@@_renderer_tl
91 \tl_new:N \l_@@_mode_tl
92 \tl_new:N \l_@@_shaper_tl
93 \tl_new:N \g_@@_defined_shapes_tl
94 \tl_new:N \g_@@_single_feat_tl
95 \tl_new:N \l_@@_basename_tl
96 \tl_new:N \g_@@_curr_series_tl
97 \tl_new:N \l_@@_curr_fontname_tl
98 \tl_new:N \l_@@_curr_bfname_tl
99 \tl_new:N \l_@@_ext_filename_tl
100 \tl_new:N \l_@@_extension_tl
101 \tl_new:N \l_@@_font_path_tl
102 \tl_new:N \l_@@_fontid_tl

```

```

103 \tl_new:N \l_@@_fontname_tl
104 \tl_new:N \l_@@_options_tl
105 \tl_new:N \l_@@_saved_fontname_tl
106 \tl_new:N \l_@@_prev_unicode_name_tl

107 \tl_new:N \g_@@_nfss_enc_tl
108 \tl_new:N \g_@@_nfss_family_tl
109 \tl_new:N \l_@@_nfss_sc_tl
110 \tl_new:N \l_@@_nfss_tl
111 \tl_new:N \l_@@_nfss_fam_tl

112 \tl_new:N \l_@@_size_tl
113 \tl_new:N \l_@@_sizedfont_tl
114 \tl_new:N \l_@@_this_font_tl
115 \tl_new:N \l_@@_ttc_index_tl
116 \tl_new:N \l_@@_smcp_shape_tl

```

EM and STRONG

```

117 \tl_new:N \l_@@_emshape_query_tl
118 \tl_new:N \l_@@_em_switch_tl
119 \tl_new:N \l_@@_strong_switch_tl

```

Scratch variables

```

120 \tl_new:N \l_@@_tmp_tl
121 \tl_new:N \l_@@_tmpa_tl
122 \tl_new:N \l_@@_tmpb_tl
123 \tl_new:N \l_@@_em_tmp_tl
124 \tl_new:N \l_@@_strong_tmp_tl

```

Maths fonts

```

125 \tl_new:N \g_@@_mathrm_tl
126 \tl_new:N \g_@@_bfmathrm_tl
127 \tl_new:N \g_@@_mathsf_tl
128 \tl_new:N \g_@@_mathtt_tl

```

Defaults: (these are set elsewhere; TODO: check if redundant)

```

129 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
130 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
131 \tl_gset:Nn \g_@@_mathtt_tl {\ttdefault}

132 \tl_new:N \l_@@_family_label_tl
133 \tl_new:N \l_@@_fake_slant_tl
134 \tl_new:N \l_@@_fake_embolden_tl

```

Internal font names

```

135 \tl_new:N \l_@@_fontname_up_tl
136 \tl_new:N \l_@@_fontname_bf_tl
137 \tl_new:N \l_@@_fontname_it_tl
138 \tl_new:N \l_@@_fontname_bfit_tl
139 \tl_new:N \l_@@_fontname_sl_tl
140 \tl_new:N \l_@@_fontname_bfsl_tl

```

```

141 \tl_new:N \l_@@_fontname_sw_tl
142 \tl_new:N \l_@@_fontname_bfsw_tl
143 \tl_new:N \l_@@_fontname_sc_tl

```

Script and Language

```

144 \tl_new:N \l_@@_script_tl
145 \tl_new:N \l_@@_script_name_tl
146 \tl_set:Nn \l_@@_script_name_tl {CustomDefault}

147 \tl_new:N \l_@@_lang_tl
148 \tl_new:N \l_@@_lang_name_tl
149 \tl_set:Nn \l_@@_lang_name_tl {Default}

```

Generic font features

```

150 \tl_new:N \l_@@_scale_tl
151 \tl_new:N \l_@@_hyphenchar_tl
152 \tl_new:N \l_@@_hexcol_tl
153 \tl_new:N \l_@@_opacity_tl
154 \tl_new:N \l_@@_optical_size_tl
155 \tl_new:N \l_@@_mapping_tl
156 \tl_new:N \l_@@_punctspace_adjust_tl
157 \tl_new:N \l_@@_wordspace_adjust_tl
158 \tl_new:N \l_@@_postadjust_tl
159 \tl_new:N \g_@@_instance_tl

160 \tl_const:Nn \c_@@_hexcol_tl {000000}
161 <XE> \tl_const:Nn \c_@@_opacity_tl {FF~}
162 <LU> \tl_const:Nn \c_@@_opacity_tl {}
163 \tl_const:Nn \c_@@_postadjust_tl { \l_@@_wordspace_adjust_tl \l_@@_punctspace_adjust_tl }

```

Semi-colon-lists Not a real data structure but sensible to name accordingly.

```

164 \tl_new:N \g_@@_rawfeatures_sclist
165 \tl_new:N \l_@@_pre_feat_sclist

```

Font families

```

166 \tl_new:N \l_@@_rmfamily_family_tl
167 \tl_new:N \l_@@_sffamily_family_tl
168 \tl_new:N \l_@@_ttfamily_family_tl
169 \tl_new:N \l_@@_rmfamily_encoding_tl
170 \tl_new:N \l_@@_sffamily_encoding_tl
171 \tl_new:N \l_@@_ttfamily_encoding_tl

```

File IV

fontspec-code-msg.dtx

1 Error/warning/info messages

Shorthands for messages:

```
1 \cs_new:Npn \@@_error:n { \msg_error:nn {fontspec} }
2 \cs_new:Npn \@@_error:nn { \msg_error:nnn {fontspec} }
3 \cs_new:Npn \@@_error:nx { \msg_error:nxx {fontspec} }
4 \cs_new:Npn \@@_error:nxx { \msg_error:nxxx {fontspec} }
5 \cs_new:Npn \@@_warning:n { \msg_warning:nn {fontspec} }
6 \cs_new:Npn \@@_warning:nx { \msg_warning:nxx {fontspec} }
7 \cs_new:Npn \@@_warning:nxx { \msg_warning:nxxx {fontspec} }
8 \cs_new:Npn \@@_info:n { \msg_info:nn {fontspec} }
9 \cs_new:Npn \@@_info:nx { \msg_info:nxx {fontspec} }
10 \cs_new:Npn \@@_info:nxx { \msg_info:nxxx {fontspec} }
11 \cs_new:Npn \@@_trace:n { \msg_trace:nn {fontspec} }
```

Allow messages to be written with spaces acting as normal:

```
12 \cs_generate_variant:Nn \msg_new:nnn {nxx}
13 \cs_generate_variant:Nn \msg_new:nnnn {nxxx}
14 \cs_new:Nn \@@_msg_new:nn
15 { \msg_new:nxx {fontspec} {#1} { \tl_trim_spaces:n {#2} } }
16 \cs_new:Nn \@@_msg_new:nnn
17 { \msg_new:nxxx {fontspec} {#1} { \tl_trim_spaces:n {#2} } { \tl_trim_spaces:n {#3} } }
18 \char_set_catcode_space:n {32}
```

1.1 Errors

```
19 \@@_msg_new:nn {only-inside-encdef}
20 {
21   \exp_not:N #1 can only be used in the second argument
22   to \string\DeclareUnicodeEncoding.
23 }
24 \@@_msg_new:nn {no-size-info}
25 {
26   Size information must be supplied.\\
27   For example, SizeFeatures={Size={8-12},...}.
28 }
29 \@@_msg_new:nnn {font-not-found}
30 {
31   The font "#1" cannot be found.
32 }
33 {
34   A font might not be found for many reasons.\\
35   Check the spelling, where the font is installed etc. etc.\\
36   When in doubt, ask someone for help!
37 }
38 \@@_msg_new:nnn {rename-feature-not-exist}
```

```

39 {
40   The feature #1 doesn't appear to be defined.
41 }
42 {
43   It looks like you're trying to rename a feature that doesn't exist.
44 }
45 \@@_msg_new:nn {no-glyph}
46 {
47   '#1' does not contain glyph #2.
48 }
49 \@@_msg_new:nnn {euler-too-late}
50 {
51   The euler package must be loaded BEFORE fontspec.
52 }
53 {
54   fontspec only overwrites euler's attempt to
55   define the maths text fonts if fontspec is
56   loaded after euler. Type <return> to proceed
57   with incorrect \string\mathit, \string\mathbf, etc.
58 }
59 \@@_msg_new:nnn {no-xcolor}
60 {
61   Cannot load named colours without the xcolor package.
62 }
63 {
64   Sorry, I can't do anything to help. Instead of loading
65   the color package, use xcolor instead.
66 }
67 \@@_msg_new:nnn {unknown-color-model}
68 {
69   Error loading colour `#1'; unknown colour model.
70 }
71 {
72   Sorry, I can't do anything to help. Please report this error
73   to my developer with a minimal example that causes the problem.
74 }
75 \@@_msg_new:nnn {not-in-addfontfeatures}
76 {
77   The "#1" font feature cannot be used in \string\addfontfeatures.
78 }
79 {
80   This is due to how TeX loads fonts; such settings
81   are global so adding them mid-document within a group causes
82   confusion. You'll need to define multiple font families to achieve
83   what you want.
84 }

```

1.2 Warnings

```

85 \@@_msg_new:nn {tu-clash}
86 {
87   I have found the tuenc.def encoding definition file but the TU encoding is not

```

```

88   defined by the LaTeX2e kernel; attempting to correct but you really should update
89   to the latest version of LaTeX2e.
90   }
91   \@@_msg_new:nn {tu-missing}
92   {
93     The TU encoding seems to be missing; please update to the latest version of LaTeX2e.
94   }
95   \@@_msg_new:nn {addfontfeatures-ignored}
96   {
97     \string\addfontfeature (s) ignored \msg_line_context;;
98     it cannot be used with a font that wasn't selected by a fontspec command.\\
99     \\
100    The current font is "\use:c{font@name}".\\
101    \int_compare:nTF { \clist_count:n {#1} = 1 }
102      { The requested feature is "#1". }
103      { The requested features are "#1". }
104    }
105   \@@_msg_new:nn {feature-option-overwrite}
106   {
107     Option '#2' of font feature '#1' overwritten.
108   }
109   \@@_msg_new:nn {ot-tag-too-long}
110   {
111     OpenType tag '#1' is too long; script, language, and feature tags must be four characters or
112   }
113   \@@_msg_new:nn {aat-feature-not-exist}
114   {
115     '\l_keys_key_tl=\l_keys_value_tl' feature not supported
116     for AAT font '\l_fontspec_fontname_tl'.
117   }
118   \@@_msg_new:nn {aat-feature-not-exist-in-font}
119   {
120     AAT feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
121     in font '\l_fontspec_fontname_tl'.
122   }
123   \@@_msg_new:nn {icu-feature-not-exist}
124   {
125     '\l_keys_key_tl=\l_keys_value_tl' feature not supported
126     for OpenType font '\l_fontspec_fontname_tl'
127   }
128   \@@_msg_new:nn {icu-feature-not-exist-in-font}
129   {
130     OpenType feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
131     for font '\l_fontspec_fontname_tl'
132     with script '\l_@@_script_name_tl' and language '\l_@@_lang_name_tl'.
133   }
134   \@@_msg_new:nn {no-opticals}
135   {
136     '#1' doesn't appear to have an Optical Size axis.
137   }
138   \@@_msg_new:nn {language-not-exist}

```

```

139 {
140   Language '#1' not available
141   for font '\l_fontspec_fontname_tl'
142   with script '\l_@@_script_name_tl'.
143 }
144 \@@_msg_new:nn {only-xetex-feature}
145 {
146   Ignored XeTeX-only feature: '#1'.
147 }
148 \@@_msg_new:nn {only-luatex-feature}
149 {
150   Ignored LuaTeX-only feature: '#1'.
151 }
152 \@@_msg_new:nn {unknown-renderer}
153 {
154   Renderer '#1' unknown. Assuming Harfbuzz with 'shaper=#1'.
155   Please raise a fontspec issue to add this shaper to the interface.
156 }
157 \@@_msg_new:nn {no-mapping}
158 {
159   Input mapping not supported in LuaTeX.
160 }
161 \@@_msg_new:nn {no-mapping-ligtext}
162 {
163   Input mapping not supported in LuaTeX.\\
164   Use "Ligatures=TeX" instead of "Mapping=tex-text".
165 }

message for package options must be loaded earlier
166 </fontspec>
167 <*options>
168 \msg_new:nnn {fontspec} {cm-default-obsolete}
169 {
170   The-"cm-default"-package-option-is-obsolete.
171 }
172 \msg_new:nnn {fontspec} {enc-obsolete}
173 {
174   The-"#1"-package-option-is-obsolete.-TU-is-the-default-encoding.
175 }
176 </options>
177 <*fontspec>
178 \@@_msg_new:nn {font-index-needs-ttc}
179 {
180   The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
181   Feature ignored.
182 }
183 \@@_msg_new:nn {feat-cannot-remove}
184 {
185   The "#1" feature cannot be deactivated. Request ignored.
186 }

```

1.3 Info messages


```

187 \@@_msg_new:nn {defining-font}
188 {
189   Font family '\g_@@_nfss_family_tl' created for font '#2'
190   with options [\l_@@_all_features_clist].\\
191   \\
192   This font family consists of the following NFSS series/shapes:\\
193   \g_@@_defined_shapes_tl
194 }
195 \@@_msg_new:nn {no-font-shape}
196 {
197   Could not resolve font "#1" (it probably doesn't exist).
198 }
199 \@@_msg_new:nn {set-scale}
200 {
201   \l_fontspeg_fontname_tl\space scale = \l_@@_scale_tl.
202 }
203 \@@_msg_new:nn {setup-math}
204 {
205   Adjusting the maths setup (use [no-math] to avoid this).
206 }
207 \@@_msg_new:nn {no-script}
208 {
209   Font "#1" does not contain requested Script "#2".
210 }
211 \@@_msg_new:nn {opa-twice}
212 {
213   Opacity set twice, in both Colour and Opacity.\\
214   Using specification "Opacity=#1".
215 }
216 \@@_msg_new:nn {opa-twice-col}
217 {
218   Opacity set twice, in both Opacity and Colour.\\
219   Using an opacity specification in hex of "#1/FF".
220 }
221 \@@_msg_new:nn {bad-colour}
222 {
223   Bad colour declaration "#1".
224   Colour must be one of:\\
225   * a named xcolor colour\\
226   * a six-digit hex colour RRGGBB\\
227   * an eight-digit hex colour RRGGBBTT with opacity
228 }

Reset 'space' behaviour:
229 \char_set_catcode_ignore:n {32}

```

File V

fontspec-code-opening.dtx

1 Opening code

1.1 Package options

```
1 \DeclareKeys
2 {
3   cm-default .code:n = { \msg_warning:nn {fontspec} {cm-default-obsolete} }
4   ,math .bool_gset:N = \g_@@_math_bool
5   ,math .usage:n = preamble
6   ,no-math .bool_gset_inverse:N = \g_@@_math_bool
7   ,no-math .usage:n = preamble
8   ,config .bool_gset:N = \g_@@_cfg_bool
9   ,config .usage:n = load
10  ,no-config .bool_gset_inverse:N = \g_@@_cfg_bool
11  ,no-config .usage:n = load
12  ,euenc .code:n = { \msg_warning:nnn {fontspec} {enc-obsolete}{euenc} }
13  ,tuenc .code:n = { \msg_warning:nnn {fontspec} {enc-obsolete}{tuenc} }
14  ,quiet .code:n =
15  {
16    \msg_redirect_module:nnn { fontspec } { warning } { info }
17    \msg_redirect_module:nnn { fontspec } { info } { none }
18  }
19  ,silent .code:n =
20  {
21    \msg_redirect_module:nnn { fontspec } { warning } { none }
22    \msg_redirect_module:nnn { fontspec } { info } { none }
23  }
24  ,verbose .code:n =
25  {
26    \msg_redirect_module:nnn { fontspec } { warning } { warning }
27    \msg_redirect_module:nnn { fontspec } { info } { info }
28  }
29 }
30 \SetKeys{config,math}
31 \ProcessKeyOptions
```

1.2 Encodings

Now the default, with a just-in-case check:

```
32 \cs_if_exist:cF {T@TU}
33 {
34   \@@_warning:n {tu-clash}
35   \DeclareFontEncoding{TU}{-}{-}
36   \DeclareFontSubstitution{TU}{lmr}{m}{n}
37 }
38 \tl_gset:Nn \g_fontspec_encoding_tl { TU }
```

```

39 \tl_set:Nn \rmdefault {lmr}
40 \tl_set:Nn \sfdefault {lmss}
41 \tl_set:Nn \ttdefault {lmtt}
42 \RequirePackage[\g_fontspec_encoding_tl]{fontenc}
43 \tl_set_eq:NN \UTFencname \g_fontspec_encoding_tl % for xunicode if needed

```

To overcome the encoding changing the current font size, but only if a class has been loaded first:

```

44 \tl_if_in:NnT \@filelist {.cls} { \normalsize }

```

Dealing with a couple of the problems introduced by babel:

```

45 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
46 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
47 \AtBeginDocument
48 {
49   \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
50   \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
51 }

```

That latin encoding definition is repeated to suppress font warnings. Something to do with `\select@language` ending up in the `.aux` file which is read at the beginning of the document.

1.3 Generic functions

`\FontspecSetCheckBoolTrue` These strange set functions are to simplify returning code from LuaTeX:

```

\FontspecSetCheckBoolFalse
52 \cs_new:Npn \FontspecSetCheckBoolTrue { \bool_set_true:N \l_@@_check_bool }
53 \cs_new:Npn \FontspecSetCheckBoolFalse { \bool_set_false:N \l_@@_check_bool }

```

(End of definition for `\FontspecSetCheckBoolTrue` and `\FontspecSetCheckBoolFalse`. These functions are documented on page ??.)

`\@@_keys_set_known:nnN`

```

54 \cs_new:Nn \@@_keys_set_known:nnN
55 {
56 <debug> \typeout{::: Keys~set:~{#1}~{#2} }
57   \keys_set_known:nnN {#1} {#2} #3
58 <debug> \typeout{::: Leftover:~{#3} }
59 }
60 \cs_generate_variant:Nn \@@_keys_set_known:nnN {nx}

```

(End of definition for `\@@_keys_set_known:nnN`. This function is documented on page ??.)

`\@@_int_mult_truncate:Nn` Missing in expl3, IMO.

```

61 \cs_new:Nn \@@_int_mult_truncate:Nn
62 {
63   \int_set:Nn #1 { \_dim_eval:w #2 #1 \_dim_eval_end: }
64 }

```

(End of definition for `\@@_int_mult_truncate:Nn`. This function is documented on page ??.)

```

\@@_lua_function:ne
\@@_lua_function:nee 65 <*LU>
\@@_lua_function:neee 66 \cs_set:Npn \@@_lua_function:ne #1#2 { \lua_now:e { fontspec.#1 ("#2")
\@@_lua_function:neeee 67 \cs_set:Npn \@@_lua_function:nee #1#2#3 { \lua_now:e { fontspec.#1 ("#2", "#3")
68 \cs_set:Npn \@@_lua_function:neee #1#2#3#4 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4")
69 \cs_set:Npn \@@_lua_function:neeee #1#2#3#4#5 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4", "#5")
70 </LU>

```

(End of definition for \@@_lua_function:ne and others. These functions are documented on page ??.)

1.4 expl3 variants

```

71 \cs_generate_variant:Nn \int_set:Nn {Nv}
72 \cs_generate_variant:Nn \keys_set:nn {nx}
73 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
74 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
75 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
76 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
77 \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
78 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
79 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
80 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
81 \cs_generate_variant:Nn \tl_if_empty_p:n {e}
82 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
83 \cs_generate_variant:Nn \tl_if_empty:nF {x}
84 \cs_generate_variant:Nn \tl_if_empty:nF {f}
85 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
86 \cs_generate_variant:Nn \tl_replace_all:Nnn {Nnx}

```

File VI

fontspec-code-fontload.dtx

1 expl3 interface for primitive font loading

\@@_primitive_font_set:Nnn

\@@_primitive_font_gset:Nnn

```
1 \cs_set:Npn \@@_primitive_font_set:Nnn #1#2#3
2 {
3   \font #1 = #2 ~at~ \dim_eval:n {#3} \scan_stop:
4 }
5 \cs_set:Npn \@@_primitive_font_gset:Nnn #1#2#3
6 {
7   \global \font #1 = #2 ~at~ \dim_eval:n {#3} \scan_stop:
8 }
```

(End of definition for \@@_primitive_font_set:Nnn and \@@_primitive_font_gset:Nnn. These functions are documented on page ??.)

\@@_font_suppress_not_found_error:

```
9 \cs_set:Npn \@@_font_suppress_not_found_error:
10 {
11   \int_set:Nn \suppressfontnotfounderror {1}
12 }
```

(End of definition for \@@_font_suppress_not_found_error:. This function is documented on page ??.)

\@@_primitive_font_if_null_p:N

\@@_primitive_font_if_null:NTF

```
13 \prg_new_conditional:Nnn \@@_primitive_font_if_null:N {p,TF,T,F}
14 {
15   \ifx #1 \nullfont
16     \prg_return_true:
17   \else
18     \prg_return_false:
19   \fi
20 }
```

(End of definition for \@@_primitive_font_if_null:NTF. This function is documented on page ??.)

\@@_primitive_font_set_p:NnnTF

\@@_primitive_font_set:NnnTFTF

\@@_primitive_font_gset_p:NnnTF

\@@_primitive_font_gset:NnnTFTF

```
21 \prg_new_conditional:Nnn \@@_primitive_font_set:Nnn {TF,T,F}
22 {
23   \@@_primitive_font_set:Nnn #1 {#2} {#3}
24   \@@_primitive_font_if_null:NTF #1 {\prg_return_false:} {\prg_return_true:}
25 }
26 \prg_new_conditional:Nnn \@@_primitive_font_gset:Nnn {TF,T,F}
27 {
28   \@@_primitive_font_gset:Nnn #1 {#2} {#3}
29   \@@_primitive_font_if_null:NTF #1 {\prg_return_false:} {\prg_return_true:}
30 }
31 \cs_set:Npn \@@_primitive_font_set:Omm { \exp_last_unbraced:No \@@_primitive_font_set:Nnn }
```

```

32 \cs_set:Npn \@@_primitive_font_set:0nnF { \exp_last_unbraced:No \@@_primitive_font_set:NnnF }
33 \cs_set:Npn \@@_primitive_font_gset:0nn { \exp_last_unbraced:No \@@_primitive_font_gset:Nnn }
34 \cs_set:Npn \@@_primitive_font_gset:0nnF { \exp_last_unbraced:No \@@_primitive_font_gset:NnnF }

```

(End of definition for \@@_primitive_font_set:NnnTTF and \@@_primitive_font_gset:NnnTTF. These functions are documented on page ??.)

\@@_primitive_font_if_exist:nTF

```

35 \prg_new_conditional:Nnn \@@_primitive_font_if_exist:n {TF,T,F}
36 {
37   \group_begin:
38   \@@_font_suppress_not_found_error:
39   \@@_primitive_font_set:Nnn \l_@@_primitive_font_{#1} { \f@size pt - 1sp }
40   \@@_primitive_font_if_null:NTF \l_@@_primitive_font
41   { \group_end: \prg_return_false: }
42   { \group_end: \prg_return_true: }
43 }

```

(End of definition for \@@_primitive_font_if_exist:nTF. This function is documented on page ??.)

\@@_primitive_font_glyph_if_exist:NnTF

```

44 \prg_new_conditional:Nnn \@@_primitive_font_glyph_if_exist:Nn {p,TF,T,F}
45 {
46   \tex_iffontchar:D #1 #2 \scan_stop:
47   \prg_return_true:
48   \else:
49   \prg_return_false:
50   \fi:
51 }

```

(End of definition for \@@_primitive_font_glyph_if_exist:NnTF. This function is documented on page ??.)

\@@_primitive_font_set_hyphenchar:Nn

```

52 \cs_new:Nn \@@_primitive_font_set_hyphenchar:Nn
53 {
54   \tex_hyphenchar:D #1 = #2 \scan_stop:
55 }

```

(End of definition for \@@_primitive_font_set_hyphenchar:Nn. This function is documented on page ??.)

\@@_primitive_font_get_name:N

\@@_primitive_font_current_name:

```

56 \cs_new_eq:NN \@@_primitive_font_get_name:N \fontname
57 \cs_new:Npn \@@_primitive_font_current_name:
58 {
59   \@@_primitive_font_get_name:N \tex_font:D
60 }

```

(End of definition for \@@_primitive_font_get_name:N and \@@_primitive_font_current_name:. These functions are documented on page ??.)

File VII

fontspec-code-interfaces.dtx

1 User commands

This section contains the definitions of the commands detailed in the user documentation. Only the 'top level' definitions of the commands are contained herein; they all use or define macros which are defined or used later on in [Section 1 on page 27](#).

```
1 \NewDocumentCommand \fontspec { 0{} m 0{} }
2 {
3   \@@_main_fontspec:nn {#1,#3} {#2}
4   \ignorespaces
5 }
6 \NewDocumentCommand \setmainfont { 0{} m 0{} }
7 {
8   \@@_main_setmainfont:nn {#1,#3} {#2}
9   \ignorespaces
10 }
11 \NewDocumentCommand \setsansfont { 0{} m 0{} }
12 {
13   \@@_main_setsansfont:nn {#1,#3} {#2}
14   \ignorespaces
15 }
16 \NewDocumentCommand \setmonofont { 0{} m 0{} }
17 {
18   \@@_main_setmonofont:nn {#1,#3} {#2}
19   \ignorespaces
20 }
21 \NewDocumentCommand \setmathrm { 0{} m 0{} }
22 {
23   \@@_main_setmathrm:nn {#1,#3} {#2}
24 }
25 \NewDocumentCommand \setboldmathrm { 0{} m 0{} }
26 {
27   \@@_main_setboldmathrm:nn {#1,#3} {#2}
28 }
29 \NewDocumentCommand \setmathsf { 0{} m 0{} }
30 {
31   \@@_main_setmathsf:nn {#1,#3} {#2}
32 }
33 \NewDocumentCommand \setmathtt { 0{} m 0{} }
34 {
35   \@@_main_setmathtt:nn {#1,#3} {#2}
36 }
```

`\setromanfont` This is the old name for `\setmainfont`, retained *ad infinitum* for backwards compatibility. It was deprecated in 2010.

```

37 \NewDocumentCommand \setromanfont { O{} m O{} }
38 {
39   \@@_main_setmainfont:nn {#1,#3} {#2}
40 }

```

(End of definition for `\setromanfont`. This function is documented on page ??.)

```

41 \NewDocumentCommand \newfontfamily { m O{} m O{} }
42 {
43   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
44 }
45 \NewDocumentCommand \renewfontfamily { m O{} m O{} }
46 {
47   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
48 }
49 \NewDocumentCommand \setfontfamily { m O{} m O{} }
50 {
51   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
52 }
53 \NewDocumentCommand \providefontfamily { m O{} m O{} }
54 {
55   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
56 }
57 \NewDocumentCommand \newfontface { m O{} m O{} }
58 {
59   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
60 }
61 \NewDocumentCommand \renewfontface { m O{} m O{} }
62 {
63   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
64 }
65 \NewDocumentCommand \setfontface { m O{} m O{} }
66 {
67   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
68 }
69 \NewDocumentCommand \providefontface { m O{} m O{} }
70 {
71   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
72 }

```

`\defaultfontfeatures` This macro takes one argument that consists of all of feature options that will be applied by default to all subsequent `\fontspec` commands.

```

73 \NewDocumentCommand \defaultfontfeatures { t+ o m }
74 {
75   \IfNoValueTF {#2}
76   { \@@_set_default_features:nn {#1} {#3} }
77   { \@@_set_font_default_features:nnn {#1} {#2} {#3} }

```



```

78     \ignorespaces
79   }

(End of definition for \defaultfontfeatures. This function is documented on page ??.)

80 \NewDocumentCommand \addfontfeatures {m}
81   {
82     \@@_main_addfontfeatures:n {#1}
83   }

84 \NewDocumentCommand \addfontfeature {m}
85   {
86     \@@_main_addfontfeatures:n {#1}
87   }

88 \NewDocumentCommand \newfontfeature {mm}
89   {
90     \@@_main_newfontfeature:nn {#1} {#2}
91   }

92 \NewDocumentCommand \newAATfeature {mmmm}
93   {
94     \@@_main_newAATfeature:nnnn {#1} {#2} {#3} {#4}
95   }

96 \NewDocumentCommand \newopentypefeature {mmm}
97   {
98     \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
99   }

```

`\newICUfeature` Deprecated.

```

100 \NewDocumentCommand \newICUfeature {mmm}
101   {
102     \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
103   }

(End of definition for \newICUfeature. This function is documented on page ??.)

104 \NewDocumentCommand \aliasfontfeature {mm}
105   {
106     \@@_main_aliasfontfeature:nn {#1} {#2}
107   }

108 \NewDocumentCommand \aliasfontfeatureoption {mmm}
109   {
110     \@@_main_aliasfontfeatureoption:nnn {#1} {#2} {#3}
111   }

```

`\newfontscript` Mostly used internally, but also possibly useful for users, to define new OpenType ‘scripts’, mapping logical names to OpenType script tags.

```

112 \NewDocumentCommand \newfontscript {mm}
113   {
114     \fontspec_new_script:nn {#1} {#2}
115   }

(End of definition for \newfontscript. This function is documented on page ??.)

```

`\newfontlanguage` Mostly used internally, but also possibly useful for users, to define new OpenType ‘languages’, mapping logical names to OpenType language tags.

```
116 \NewDocumentCommand \newfontlanguage {mm}
117   {
118     \fontspec_new_lang:nn {#1} {#2}
119   }
```

(End of definition for \newfontlanguage. This function is documented on page ??.)

```
120 \NewDocumentCommand \DeclareFontExtensions {m}
121   {
122     \@@_main_DeclareFontExtensions:n {#1}
123   }
124 \NewDocumentCommand \IfFontFeatureActiveTF {mmm}
125   {
126     \@@_main_IfFontFeatureActiveTF:nnn {#1} {#2} {#3}
127   }
```

`\oldstylenums` This is performed only after the preamble to overwrite any redefinition by textcomp:

```
128 \AtBeginDocument
129   {
130     \RenewDocumentCommand \oldstylenums {m}
131       {
132         \@@_main_oldstylenums:n {#1}
133       }
134   }
```

(End of definition for \oldstylenums. This function is documented on page ??.)

`\liningnums`

```
135 \NewDocumentCommand \liningnums {m}
136   {
137     \@@_main_liningnums:n {#1}
138   }
```

(End of definition for \liningnums. This function is documented on page ??.)

File VIII

fontspec-code-user.dtx

1 User command internals

1.1 Font selection

`\@@_main_fontspec:nn` This is the main command of the package that selects fonts with various features. It takes two arguments: the font name and the optional requested features of that font.

```
1 \cs_new:Nn \@@_main_fontspec:nn
2   {
3     \fontspec_set_family:Nnn \f@family {#1} {#2}
4     \fontencoding { \g_@@_nfss_enc_tl }
5     \selectfont
6   }
```

(End of definition for \@@_main_fontspec:nn. This function is documented on page ??.)

`\rmfamily` Add an encoding switch to the three family commands.

```
\sffamily
\ttfamily
7 \cs_if_exist:NTF \@rmfamilyhook
8   {
9     \tl_put_right:Nn \@rmfamilyhook {\fontencoding \l_@@_rmfamily_encoding_tl}
10    \tl_put_right:Nn \@sffamilyhook {\fontencoding \l_@@_sffamily_encoding_tl}
11    \tl_put_right:Nn \@ttfamilyhook {\fontencoding \l_@@_ttfamily_encoding_tl}
12  }
13  {
14    \tl_replace_all:cnn { rmfamily~ } { \fontfamily }
15    { \fontencoding \l_@@_rmfamily_encoding_tl \fontfamily }
16    \tl_replace_all:cnn { sffamily~ } { \fontfamily }
17    { \fontencoding \l_@@_sffamily_encoding_tl \fontfamily }
18    \tl_replace_all:cnn { ttfamily~ } { \fontfamily }
19    { \fontencoding \l_@@_ttfamily_encoding_tl \fontfamily }
20  }
21 \tl_set:Nn \l_@@_rmfamily_encoding_tl { \encodingdefault }
22 \tl_set:Nn \l_@@_sffamily_encoding_tl { \encodingdefault }
23 \tl_set:Nn \l_@@_ttfamily_encoding_tl { \encodingdefault }
```

(End of definition for \rmfamily, \sffamily, and \ttfamily. These functions are documented on page ??.)

`\setmainfont` The following three macros perform equivalent operations setting the default font for a particular family: ‘roman’, sans serif, or typewriter (monospaced).

They end with `\normalfont` so that if they’re used in the document, the change registers immediately.

```
24 \cs_new:Nn \@@_main_setmainfont:nn
25   {
26     \ifdefined\DeclareFontSeriesDefault
27       \DeclareFontSeriesDefault[rm]{bf}{\bfdefault}
28     \fi
29     \fontspec_set_family:Nnn \l_@@_rmfamily_family_tl {#1} {#2}
30     \tl_set_eq:NN \rmdefault \l_@@_rmfamily_family_tl
```

```

31 \tl_set_eq:NN \l_@@_rmfamily_encoding_tl \g_@@_nfss_enc_tl
32 \str_if_eq:eeT {\familydefault} {\rmdefault}
33 { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
34 \@@_setmainfont_hook:nn {#1} {#2} % for unicode-math only
35 \normalfont
36 }

```

(End of definition for `\setmainfont`. This function is documented on page ??.)

`\setsansfont` Same as above.

```

37 \cs_new:Nn \@@_main_setsansfont:nn
38 {
39   \ifdefined\DeclareFontSeriesDefault
40     \DeclareFontSeriesDefault[sf]{bf}{\bfdefault}
41     \fi
42   \fontspec_set_family:Nnn \l_@@_sffamily_family_tl {#1} {#2}
43   \tl_set_eq:NN \sfdefault \l_@@_sffamily_family_tl
44   \tl_set_eq:NN \l_@@_sffamily_encoding_tl \g_@@_nfss_enc_tl
45   \str_if_eq:eeT {\familydefault} {\sfdefault}
46   { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
47   \@@_setsansfont_hook:nn {#1} {#2} % for unicode-math only
48   \normalfont
49 }

```

(End of definition for `\setsansfont`. This function is documented on page ??.)

`\setmonofont` Same as above.

```

50 \cs_new:Nn \@@_main_setmonofont:nn
51 {
52   \ifdefined\DeclareFontSeriesDefault
53     \DeclareFontSeriesDefault[tt]{bf}{\bfdefault}
54     \fi
55   \fontspec_set_family:Nnn \l_@@_ttfamily_family_tl {#1} {#2}
56   \tl_set_eq:NN \ttdefault \l_@@_ttfamily_family_tl
57   \tl_set_eq:NN \l_@@_ttfamily_encoding_tl \g_@@_nfss_enc_tl
58   \str_if_eq:eeT {\familydefault} {\ttdefault}
59   { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
60   \@@_setmonofont_hook:nn {#1} {#2} % for unicode-math only
61   \normalfont
62 }

```

(End of definition for `\setmonofont`. This function is documented on page ??.)

`\setmathrm` These commands are analogous to `\setmainfont` and others, but for selecting the font used for `\mathrm`, etc. They can only be used in the preamble of the document. `\setboldmathrm` is used for specifying which fonts should be used in `\boldmath`.

```

63 \cs_new:Nn \@@_main_setmathrm:nn
64 {
65 (XE) \fontspec_gset_family:Nnn \g_@@_mathrm_tl {#1} {#2}
66 (LU) \fontspec_gset_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
67   \@@_setmathrm_hook:nn {#1} {#2} % for unicode-math only
68 }

```

(End of definition for `\setmathrm`. This function is documented on page ??.)

`\setboldmathrm`

```
69 \cs_new:Nn \@@_main_setboldmathrm:nn
70 {
71 (XE) \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {#1} {#2}
72 (LU) \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {Renderer=Basic,#1} {#2}
73 \@@_setboldmathrm_hook:nn {#1} {#2} % for unicode-math only
74 }
```

(End of definition for `\setboldmathrm`. This function is documented on page ??.)

`\setmathsf`

```
75 \cs_new:Nn \@@_main_setmathsf:nn
76 {
77 (XE) \fontspec_gset_family:Nnn \g_@@_mathsf_tl {#1} {#2}
78 (LU) \fontspec_gset_family:Nnn \g_@@_mathsf_tl {Renderer=Basic,#1} {#2}
79 \@@_setmathsf_hook:nn {#1} {#2} % for unicode-math only
80 }
```

(End of definition for `\setmathsf`. This function is documented on page ??.)

`\setmathtt`

```
81 \cs_new:Nn \@@_main_setmathtt:nn
82 {
83 (XE) \fontspec_gset_family:Nnn \g_@@_mathtt_tl {#1} {#2}
84 (LU) \fontspec_gset_family:Nnn \g_@@_mathtt_tl {Renderer=Basic,#1} {#2}
85 \@@_setmathtt_hook:nn {#1} {#2} % for unicode-math only
86 }
```

(End of definition for `\setmathtt`. This function is documented on page ??.)

Hooks:

```
87 \cs_set_eq:NN \@@_setmainfont_hook:nn \use_none:nn
88 \cs_set_eq:NN \@@_setsansfont_hook:nn \use_none:nn
89 \cs_set_eq:NN \@@_setmonofont_hook:nn \use_none:nn
90 \cs_set_eq:NN \@@_setmathrm_hook:nn \use_none:nn
91 \cs_set_eq:NN \@@_setmathsf_hook:nn \use_none:nn
92 \cs_set_eq:NN \@@_setmathtt_hook:nn \use_none:nn
93 \cs_set_eq:NN \@@_setboldmathrm_hook:nn \use_none:nn
```

Hmm, this isn't necessary with unicode-math; oh well:

```
94 \@onlypreamble\setmathrm
95 \@onlypreamble\setboldmathrm
96 \@onlypreamble\setmathsf
97 \@onlypreamble\setmathtt
```

If the commands above are not executed, then `\rmdefault` (*etc.*) will be used.

```
98 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
99 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
100 \tl_gset:Nn \g_@@_mathtt_tl {\ttdefault}
```

`\@@_main_newfontfamily:NnnN` The inner fontspec workings define a font family, which is then used in a typical NFSS `\fontfamily` declaration, saved in the macro name specified. The fourth argument determines which xparse function to set the macro with (new/renew/etc).

```

101 \cs_new:Nn \@@_main_newfontfamily:NnnN
102   {
103     \fontspec_set_family:cnn { l_@@_ \cs_to_str:N #1 _family_tl } {#2} {#3}
104     \use:x
105     {
106       \exp_not:N #4 \exp_not:N #1 {}
107       {
108         \exp_not:N \fontfamily { \use:c { l_@@_ \cs_to_str:N #1 _family_tl } }
109         \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
110         \exp_not:N \selectfont
111       }
112     }
113   }

```

(End of definition for \@@_main_newfontfamily:NnnN. This function is documented on page ??.)

`\@@_main_newfontface:NnnN` `\newfontface` uses the fact that if the argument to `BoldFont`, etc., is empty (*i.e.*, `BoldFont={}`), then no bold font is searched for.

```

114 \cs_new:Nn \@@_main_newfontface:NnnN
115   {
116     \@@_main_newfontfamily:NnnN #1 { BoldFont={},ItalicFont={},SmallCapsFont={},#2 } {#3} #4
117   }

```

(End of definition for \@@_main_newfontface:NnnN. This function is documented on page ??.)

1.2 Font feature selection

`\@@_set_default_features:nn`

```

118 \cs_new:Nn \@@_set_default_features:nn
119   {
120     \IfBooleanTF {#1} \clist_gput_right:Nn \clist_gset:Nn
121       \g_@@_default_fontopts_clist {#2}
122   }

```

(End of definition for \@@_set_default_features:nn. This function is documented on page ??.)

`\@@_set_font_default_features:nnn` The optional argument #2 specifies font identifier(s). Branch for either (a) single token input such as `\rmdefault`, or (b) otherwise assume its a fontname. In that case, strip spaces and file extensions and lower-case to ensure consistency.

```

123 \cs_new:Nn \@@_set_font_default_features:nnn
124   {
125     <debug> \typeout{\unexpanded[_set_font_default_features:nnn:{#1}{#2}{#3}}
126     \clist_map_inline:nn {#2}
127     {
128       \tl_if_single:nTF {##1}
129         { \tl_set:No \l_@@_tmp_tl { \cs:w l_@@_ \cs_to_str:N ##1 _family_tl\cs_end: } }
130         { \@@_sanitize_fontname:Nn \l_@@_tmp_tl {##1} }
131     }

```

```

132     \IfBooleanTF {#1}
133     {
134         \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
135         { \tl_clear:N \l_@@_tmpb_tl }
136         \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
137         \prop_gput:NVV \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
138     }
139     {
140         \tl_if_empty:nTF {#3}
141         { \prop_gremove:NV \g_@@_fontopts_prop \l_@@_tmp_tl }
142         { \prop_gput:NVn \g_@@_fontopts_prop \l_@@_tmp_tl {#3,} }
143     }
144 }
145 }

```

(End of definition for `\@@_set_font_default_features:nnn`. This function is documented on page ??.)

`\addfontfeatures` In order to be able to extend the feature selection of a given font, two things need to be known: the currently selected features, and the currently selected font. Every time a font family is created, this information is saved inside a control sequence with the name of the font family itself.

This macro extracts this information, then appends the requested font features to add to the already existing ones, and calls the font again with the top level `\fontspec` command.

The default options are *not* applied (which is why `\g_fontspec_default_fontopts_tl` is emptied inside the group; this is allowed as `\l_fontspec_family_tl` is globally defined in `\@@_select_font_family:nn`), so this means that the only added features to the font are strictly those specified by this command.

`\addfontfeature` is defined as an alias, as I found that I often typed this instead when adding only a single font feature.

```

146 \cs_new:Nn \@@_main_addfontfeatures:n
147 {
148 <debug> \typeout{^^J:.....:^^J: addfontfeatures}
149   \fontspec_if_fontspec_font:TF
150   {
151     \group_begin:
152     \keys_set_known:nnN {fontspec-addfeatures} {#1} \l_@@_tmp_tl
153     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {options} \l_@@_options_tl
154     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {fontname} \l_@@_fontname_tl
155     \bool_set_true:N \l_@@_disable_defaults_bool
156 <debug> \typeout{ \@@_select_font_family:nn { \l_@@_options_tl , #1 } {\l_@@_fontname_tl} }
157     \use:x
158     {
159       \@@_select_font_family:nn
160       { \l_@@_options_tl , #1 } {\l_@@_fontname_tl}
161     }
162     \group_end:
163     \fontfamily \g_@@_nfss_family_tl \selectfont
164   }
165   {
166     \@@_warning:nx {addfontfeatures-ignored} {#1}
167   }

```

```

168     \ignorespaces
169   }

```

(End of definition for \addfontfeatures. This function is documented on page ??.)

1.3 Defining new font features

`\newfontfeature` `\newfontfeature` takes two arguments: the name of the feature tag by which to reference it, and the string that is used to select the font feature.

```

170 \cs_new:Nn \@@_main_newfontfeature:nn
171   {
172     \keys_define:nn { fontspec }
173       {
174         #1 .code:n = { \@@_update_featstr:n {#2} }
175       }
176   }

```

(End of definition for \newfontfeature. This function is documented on page ??.)

`\newAATfeature` This command assigns a new AAT feature by its code (`#2`,`#3`) to a new name (`#1`). Better than `\newfontfeature` because it checks if the feature exists in the font it's being used for.

```

177 \cs_new:Nn \@@_main_newAATfeature:nnnn
178   {
179     \keys_if_exist:nnF { fontspec } {#1}
180     { \@@_define_aat_feature_group:n {#1} }
181
182     \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
183     { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
184
185     \@@_define_aat_feature:nnnn {#1}{#2}{#3}{#4}
186   }

```

(End of definition for \newAATfeature. This function is documented on page ??.)

`\newopentypefeature` This command assigns a new OpenType feature by its abbreviation (`#2`) to a new name (`#1`). Better than `\newfontfeature` because it checks if the feature exists in the font it's being used for.

```

187 \cs_new:Nn \@@_main_newopentypefeature:nnn
188   {
189     \keys_if_exist:nnF { fontspec / options } {#1}
190     { \@@_define_opentype_feature_group:n {#1} }
191
192     \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
193     { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
194
195     \exp_args:Nnnx \@@_define_opentype_feature:nnnnn
196     {#1} {#2} { \@@_strip_plus_minus:n {#3} } {#3} {}
197   }

```



```

198 \cs_new:Nn \@@_strip_plus_minus:n { \@@_strip_plus_minus_aux:Nq #1 \q_nil }
199 \cs_new:Npn \@@_strip_plus_minus_aux:Nq #1#2 \q_nil
200 {
201   \str_case:nnF {#1} { {+} {#2} {-} {#2} } {#1#2}
202 }

```

(End of definition for \newopentypefeature. This function is documented on page ??.)

\aliasfontfeature User commands for renaming font features and font feature options.

```

203 \cs_new:Nn \@@_main_aliasfontfeature:nn
204 {
205   (debug) \typeout{::::::::::::::::::::::::::^J:: aliasfontfeature{#1}{#2}}
206   \bool_set_false:N \l_@@_alias_bool
207
208   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
209   {
210     \keys_if_exist:nnT {##1} {#1}
211     {
212       (debug) \typeout{::: Key~exists~##1~/~#1}
213       \bool_set_true:N \l_@@_alias_bool
214       \keys_define:nn {##1}
215         { #2 .code:n = { \keys_set:nn {##1} { #1 = {####1} } } }
216     }
217   }
218
219   \bool_if:NF \l_@@_alias_bool
220   { \@@_warning:nx {rename-feature-not-exist} {#1} }
221 }

```

(End of definition for \aliasfontfeature. This function is documented on page ??.)

\aliasfontfeatureoption

```

222 \cs_new:Nn \@@_main_aliasfontfeatureoption:nnn
223 {
224   \bool_set_false:N \l_@@_alias_bool
225
226   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
227   {
228     \keys_if_exist:nnT { ##1 / #1 } {#2}
229     {
230       (debug) \typeout{::: Keyval~exists~##1~/~#1~~#2}
231       \bool_set_true:N \l_@@_alias_bool
232       \keys_define:nn { ##1 / #1 }
233         { #3 .code:n = { \keys_set:nn {##1} { #1 = {#2} } } }
234     }
235
236     \keys_if_exist:nnT { ##1 / #1 } {#2Reset}
237     {
238       (debug) \typeout{::: Keyval~exists~##1~/~#1~~#2Reset}
239       \keys_define:nn { ##1 / #1 }
240         { #3Reset .code:n = { \keys_set:nn {##1} { #1 = {#2Reset} } } }
241     }

```

```

242
243     \keys_if_exist:nnT { ##1 / #1 } {#20ff}
244     {
245 (debug) \typeout{::: Keyval~exists~##1~/~#1~=#20ff}
246         \keys_define:nn { ##1 / #1 }
247             { #30ff .code:n = { \keys_set:nn {##1} { #1 = {#20ff} } } }
248     }
249 }
250
251 \bool_if:NF \l_@@_alias_bool
252 { \@@_warning:nx {rename-feature-not-exist} {#1/#2} }
253 }

```

(End of definition for \aliasfontfeatureoption. This function is documented on page ??.)

\@@_main_DeclareFontExtensions:n

```

254 \cs_new:Nn \@@_main_DeclareFontExtensions:n
255 {
256     \clist_set:Nn \l_@@_extensions_clist { #1 }
257 }

```

Defaults:

```

258 \@@_main_DeclareFontExtensions:n { .otf, .ttf, .OTF, .TTF, .ttc, .TTC, .dfont }

```

(End of definition for \@@_main_DeclareFontExtensions:n. This function is documented on page ??.)

1.4 High level conditionals

\IfFontFeatureActiveTF

```

259 \cs_new:Nn \@@_main_IfFontFeatureActiveTF:nnn
260 {
261 (debug) \typeout{^^J::::::::::::::::::::::::::::::::::::::::::::::::::}
262 (debug) \typeout{:IfFontFeatureActiveTF \exp_not:n{#{#1}{#2}{#3}}}
263     \@@_if_font_feature:nTF {#1} {#2} {#3}
264 }
265 \prg_new_conditional:Nnn \@@_if_font_feature:n {TF}
266 {
267     \tl_gclear:N \g_@@_single_feat_tl
268     \group_begin:
269         \@@_font_suppress_not_found_error:
270         \@@_init:
271         \bool_set_true:N \l_@@_ot_bool
272         \bool_set_true:N \l_@@_never_check_bool
273         \bool_set_false:N \l_@@_firsttime_bool
274         \clist_clear:N \l_@@_fontfeat_clist
275         \@@_get_features:n {#1}
276     \group_end:
277
278 (debug) \typeout{::: > \exp_not:N\g_@@_rawfeatures_sclist->~{\g_@@_rawfeatures_sclist}}
279 (debug) \typeout{::: > \exp_not:N\g_@@_single_feat_tl->~{\g_@@_single_feat_tl}}
280
281     \tl_if_empty:NTF \g_@@_single_feat_tl { \prg_return_false: }

```

```

282     {
283     \exp_args:NV \fontspec_if_current_feature:nTF \g_@@_single_feat_tl
284     { \prg_return_true: } { \prg_return_false: }
285     }
286 }

```

(End of definition for \IfFontFeatureActiveTF. This function is documented on page ??.)

1.5 \oldstylenums and \liningnums

`\oldstylenums` This command needs a redefinition. And we may as well provide the reverse command.

```

\liningnums 287 \cs_new_protected:Nn \@@_main_oldstylenums:n
288 {
289   \group_begin:
290   \addfontfeature{Numbers=OldStyle}
291   #1
292   \group_end:
293 }
294 \cs_new_protected:Nn \@@_main_liningnums:n
295 {
296   \group_begin:
297   \addfontfeature{Numbers=Lining}
298   #1
299   \group_end:
300 }

```

(End of definition for \oldstylenums and \liningnums. These functions are documented on page ??.)

File IX

fontspec-code-api.dtx

1 Programmer's interface

These functions are not used directly by fontspec when defining fonts; they are designed to be used by other packages who wish to do font-related things on top of fontspec itself.

Because I haven't fully explored how these functions will behave in practise, I am not giving them user-level names. As it becomes more clear which of these should be accessible by document writers, I'll open them up a little more.

All functions are defined assuming that the font to be queried is currently selected as a fontspec font. (I.e., via `\fontspec` or from a `\newfontfamily` macro or from `\setmainfont` and so on.)

```
\fontspec_if_fontspec_font:TF Test whether the currently selected font has been loaded by fontspec.
1 \prg_new_conditional:Nnn \fontspec_if_fontspec_font: {TF,T,F}
2 {
3   \cs_if_exist:cTF {g_@@_fontinfo_ \f@family _prop} \prg_return_true: \prg_return_false:
4 }
```

(End of definition for `\fontspec_if_fontspec_font:TF`. This function is documented on page ??.)

```
\fontspec_if_aat_feature:nnTF Conditional to test if the currently selected font contains the AAT feature (#1,#2).
5 \prg_new_conditional:Nnn \fontspec_if_aat_feature:nn {TF,T,F}
6 {
7   \fontspec_if_fontspec_font:TF
8   {
9     \@@_set_font_type:N \font
10    \bool_if:NTF \l_@@_atsui_bool
11    {
12      \@@_make_AAT_feature_string:NnnTF \font {#1} {#2}
13      \prg_return_true: \prg_return_false:
14    }
15    {
16      \prg_return_false:
17    }
18  }
19  {
20    \prg_return_false:
21  }
22 }
```

(End of definition for `\fontspec_if_aat_feature:nnTF`. This function is documented on page ??.)

```
\fontspec_if_opentype:TF Test whether the currently selected font is an OpenType font. Always true for LuaTeX fonts.
23 \prg_new_conditional:Nnn \fontspec_if_opentype: {TF,T,F}
24 {
25   \fontspec_if_fontspec_font:TF
26   {
```

```

27     \@@_set_font_type:N \font
28     \bool_if:NTF \l_@@_ot_bool \prg_return_true: \prg_return_false:
29   }
30   {
31     \prg_return_false:
32   }
33 }

```

(End of definition for \fontspec_if_opentype:TF. This function is documented on page ??.)

\fontspec_if_feature:nTF Test whether the currently selected font contains the raw OpenType feature #1. E.g.: `\fontspec_if_feature:nTF {TF,T,F}` Returns false if the font is not loaded by fontspec or is not an OpenType font.

```

34 \prg_new_conditional:Nnn \fontspec_if_feature:n {TF,T,F}
35 {
36   \fontspec_if_fontspec_font:TF
37   {
38     \@@_set_font_type:N \font
39     \bool_if:NTF \l_@@_ot_bool
40     {
41       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-num} \l_@@_tmp_tl
42       \int_set:Nn \l_@@_script_int {\l_@@_tmp_tl}
43
44       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-num} \l_@@_tmp_tl
45       \int_set:Nn \l_@@_language_int {\l_@@_tmp_tl}
46
47       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag} \l_@@_script_tl
48       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag} \l_@@_lang_tl
49
50       \@@_check_ot_feat:NnTF \font {#1} {\prg_return_true:} {\prg_return_false:}
51     }
52     {
53       \prg_return_false:
54     }
55   }
56   {
57     \prg_return_false:
58   }
59 }

```

(End of definition for \fontspec_if_feature:nTF. This function is documented on page ??.)

\fontspec_if_feature:nnnTF Test whether the currently selected font with raw OpenType script tag #1 and raw OpenType language tag #2 contains the raw OpenType feature tag #3. E.g.: `\fontspec_if_feature:nTF {latn} {ROM} {pnum} {True} {False}` Returns false if the font is not loaded by fontspec or is not an OpenType font.

```

60 \prg_new_conditional:Nnn \fontspec_if_feature:nnn {TF,T,F}
61 {
62   \fontspec_if_fontspec_font:TF
63   {
64     \@@_set_font_type:N \font
65     \bool_if:NTF \l_@@_ot_bool
66     {

```

```

67         \@@_check_ot_feat:NnnnTF \font {#3} {#2} {#1} \prg_return_true: \prg_return_false:
68     }
69     { \prg_return_false: }
70 }
71 { \prg_return_false: }
72 }

```

(End of definition for \fontspec_if_feature:nnnTF. This function is documented on page ??.)

\fontspec_if_script:nTF Test whether the currently selected font contains the raw OpenType script #1. E.g.: `\fontspec_if_script:nTF {ROM} {True} {False}`. Returns false if the font is not loaded by fontspec or is not an OpenType font.

```

73 \prg_new_conditional:Nnn \fontspec_if_script:n {TF,T,F}
74 {
75     \fontspec_if_fontspec_font:TF
76     {
77         \@@_set_font_type:N \font
78         \bool_if:NTF \l_@@_ot_bool
79         {
80             \@@_check_script:NnTF \font {#1} \prg_return_true: \prg_return_false:
81         }
82         { \prg_return_false: }
83     }
84     { \prg_return_false: }
85 }

```

(End of definition for \fontspec_if_script:nTF. This function is documented on page ??.)

\fontspec_if_language:nTF Test whether the currently selected font contains the raw OpenType language tag #1. E.g.: `\fontspec_if_language:nTF {ROM} {True} {False}`. Returns false if the font is not loaded by fontspec or is not an OpenType font.

```

86 \prg_new_conditional:Nnn \fontspec_if_language:n {TF,T,F}
87 {
88     \fontspec_if_fontspec_font:TF
89     {
90         \@@_set_font_type:N \font
91         \bool_if:NTF \l_@@_ot_bool
92         {
93             \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-num} \l_@@_tmp_tl
94             \int_set:Nn \l_@@_script_int {\l_@@_tmp_tl}
95             \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag} \l_@@_script_tl
96
97             \@@_check_lang:NnTF \font {#1} \prg_return_true: \prg_return_false:
98         }
99         { \prg_return_false: }
100     }
101     { \prg_return_false: }
102 }

```

(End of definition for \fontspec_if_language:nTF. This function is documented on page ??.)

`\fontspec_if_language:nnTF` Test whether the currently selected font contains the raw OpenType language tag #2 in script #1. E.g.: `\fontspec_if_language:nnTF {cyr1} {SRB} {True} {False}`. Returns false if the font is not loaded by fontspec or is not an OpenType font.

```

103 \prg_new_conditional:Nnn \fontspec_if_language:nn {TF,T,F}
104   {
105     \fontspec_if_fontspec_font:TF
106     {
107       \@@_set_font_type:N \font
108       \bool_if:NTF \l_@@_ot_bool
109       {
110         \@@_check_lang:NnnTF \font {#2} {#1} \prg_return_true: \prg_return_false:
111       }
112       { \prg_return_false: }
113     }
114     { \prg_return_false: }
115   }

```

(End of definition for `\fontspec_if_language:nnTF`. This function is documented on page ??.)

`\fontspec_if_current_script:nTF` Test whether the currently loaded font is using the specified raw OpenType script tag #1.

```

116 \prg_new_conditional:Nnn \fontspec_if_current_script:n {TF,T,F}
117   {
118     \fontspec_if_fontspec_font:TF
119     {
120       \@@_set_font_type:N \font
121       \bool_if:NTF \l_@@_ot_bool
122       {
123         \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag} \l_@@_tmp_tl
124         \str_if_eq:nVTF {#1} \l_@@_tmp_tl
125         {\prg_return_true:} {\prg_return_false:}
126       }
127       { \prg_return_false: }
128     }
129     { \prg_return_false: }
130   }

```

(End of definition for `\fontspec_if_current_script:nTF`. This function is documented on page ??.)

`\fontspec_if_current_language:nTF` Test whether the currently loaded font is using the specified raw OpenType language tag #1.

```

131 \prg_new_conditional:Nnn \fontspec_if_current_language:n {TF,T,F}
132   {
133     \fontspec_if_fontspec_font:TF
134     {
135       \@@_set_font_type:N \font
136       \bool_if:NTF \l_@@_ot_bool
137       {
138         \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag} \l_@@_tmp_tl
139         \str_if_eq:nVTF {#1} \l_@@_tmp_tl
140         {\prg_return_true:} {\prg_return_false:}
141       }
142       { \prg_return_false: }
143     }

```

```

144     { \prg_return_false: }
145   }

```

(End of definition for `\fontspec_if_current_language:nTF`. This function is documented on page ??.)

```

\fontspec_set_family:Nnn #1 : family
                        #2 : fontspec features
                        #3 : font name

```

Defines a new font family from given *features* and *font*, and stores the name in the variable *family*. See the standard fontspec user commands for applications of this function.

We want to store the actual name of the font family within the *family* variable because the actual L^AT_EX family name is automatically generated by fontspec and it's easier to keep it that way.

```

146 \cs_new:Nn \@@_tl_new_if_free:N { \tl_if_exist:NF #1 { \tl_new:N #1 } }
147 \cs_new:Nn \@@_set_family:NnnN
148   {
149     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
150     \tl_set:Nn \l_@@_family_label_tl {#1}
151     \@@_select_font_family:nn {#2} {#3}
152     \@@_tl_new_if_free:N #1
153     #4 #1 \l_fontspec_family_tl
154     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
155   }
156 \cs_new:Nn \fontspec_gset_family:Nnn { \@@_set_family:NnnN #1 {#2} {#3} \tl_gset_eq:NN }
157 \cs_new:Nn \fontspec_set_family:Nnn { \@@_set_family:NnnN #1 {#2} {#3} \tl_set_eq:NN }
158 \cs_generate_variant:Nn \fontspec_set_family:Nnn {c}

```

(End of definition for `\fontspec_set_family:Nnn`. This function is documented on page ??.)

```

\fontspec_set_fontface:NNnnn TODO: the round-about approach of using \fontname means that settings such as fontdi-
                             mens will be lost. (Discovered in unicode-math.) Investigate!

```

```

159 \tl_new:N \l_@@_fontface_cs_tl
160 \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font}
161 \cs_new:Nn \@@_set_fontface:NNnnN
162   {
163     \tl_set:Nn \l_@@_fontface_cs_tl {#1}
164     \tl_set:Nn \l_@@_family_label_tl {#2}
165     \@@_select_font_family:nn {#3} {#4}
166     #5 #2 \l_fontspec_family_tl
167     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
168   }
169 \cs_new:Nn \fontspec_gset_fontface:NNnnn { \@@_set_fontface:NNnnN #1 #2 {#3} {#4} \tl_gset_eq:NN }
170 \cs_new:Nn \fontspec_set_fontface:NNnnn { \@@_set_fontface:NNnnN #1 #2 {#3} {#4} \tl_set_eq:NN }

```

(End of definition for `\fontspec_set_fontface:NNnn`. This function is documented on page ??.)

```

\fontspec_font_if_exist:n

```

```

171 \prg_new_conditional:Nnn \fontspec_font_if_exist:n {TF,T,F}
172   {
173     \group_begin:
174     \@@_init:

```



```

175     \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
176     \@@_primitive_font_if_exist:nTF { \@@_construct_font_call:nn {#1} {} }
177         { \group_end: \prg_return_true: }
178         { \group_end: \prg_return_false: }
179     }

```

```

180 \cs_set_eq:NN \IfFontExistsTF \fontspec_font_if_exist:nTF

```

(End of definition for \fontspec_font_if_exist:n. This function is documented on page ??.)

`\fontspec_if_current_feature:nTF` Test whether the currently loaded font is using the specified raw OpenType feature tag #1.

```

181 \prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
182 {
183   <debug>\typeout{::~fontspec_if_current_feature:n~{#1}}
184   <debug>\typeout{::::~primitive_font_current_name::~~\@@_primitive_font_current_name:}
185   \exp_args:Nxx \tl_if_in:nnTF
186     { \@@_primitive_font_current_name: } { \tl_to_str:n {#1} }
187     { \prg_return_true: } { \prg_return_false: }
188 }

```

(End of definition for \fontspec_if_current_feature:nTF. This function is documented on page ??.)

`\fontspec_if_small_caps:TF`

```

189 \prg_new_conditional:Nnn \fontspec_if_small_caps: {TF,T,F}
190 {
191   \@@_if_merge_shape:nTF {sc}
192     {
193       \tl_set_eq:Nc \l_@@_smcp_shape_tl { \@@_shape_merge:nn {\f@shape} {sc} }
194     }
195     {
196       \tl_set:Nn \l_@@_smcp_shape_tl {sc}
197     }
198
199   \cs_if_exist:cTF { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
200     {
201       \tl_if_eq:ccTF
202         { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
203         { \f@encoding/\f@family/\f@series/\shapedefault }
204         { \prg_return_false: }
205         { \prg_return_true: }
206     }
207     { \prg_return_false: }
208 }

```

(End of definition for \fontspec_if_small_caps:TF. This function is documented on page ??.)

File X

fontspec-code-internal.dtx

1 Internals

1.1 The main function for setting fonts

`\@@_select_font_family:nn` This is the command that defines font families for use, the underlying procedure of all `\fontspec`-like commands. Given a list of font features (`#1`) for a requested font (`#2`), it will define an NFSS family for that font and put the family name (globally) into `\l_fontspec_family_tl`. The TeX `\font` command is (globally) stored in `\l_fontspec_font`.

This macro does its processing inside a group to attempt to restrict the scope of its internal processing. This works to some degree to insulate the internal commands from having to be manually cleared.

Some often-used variables to know about:

- `\l_fontspec_fontname_tl` is used as the generic name of the font being defined.
- `\l_@@_fontid_tl` is the unique identifier of the font with all its features.
- `\l_@@_fontname_up_tl` is the font specifically to be used as the upright font.
- `\l_@@_basename_tl` is the (immutable) original argument used for `*`-replacing.
- `\l_fontspec_font` is the plain TeX font of the upright font requested.

```
1 \cs_new_protected:Nn \@@_select_font_family:nn
2 {
3   (debug)\typeout{^^J^^J::::::::::::::::::::::::::::::::::::::::^^J:: fontspec_select:nn~ {#1}~ {#2} }
4   \group_begin:
5   \@@_font_suppress_not_found_error:
6   \@@_init:
7
8   \@@_sanitise_fontname:Nn \l_fontspec_fontname_tl {#2}
9   \@@_sanitise_fontname:Nn \l_@@_fontname_up_tl {#2}
10  \@@_sanitise_fontname:Nn \l_@@_basename_tl {#2}
11
12  \@@_if_detect_external:nT {#2}
13    { \keys_set:nn {fontspec-preparse-external} {Path} }
14
15  \keys_set_known:nn {fontspec-preparse-cfg} {#1}
16
17  \@@_init_ttc:n {#2}
18  \@@_load_external_fontoptions:N \l_fontspec_fontname_tl
19
20  \@@_extract_all_features:n {#1}
21  \tl_set:Nx \l_@@_fontid_tl { \tl_to_str:N \l_fontspec_fontname_tl-:\tl_to_str:N \l_@@_all
22
23  (debug)\typeout{fontid: \l_@@_fontid_tl}
24
25  \@@_preparse_features:
```

```

26 \@@_load_font:
27 \@@_set_scriptlang:
28 \@@_get_features:n {}
29 \bool_set_false:N \l_@@_firsttime_bool
30
31 \@@_save_family_needed:nTF {#2}
32 {
33   \@@_save_family:nn {#1} {#2}
34 <debug>\@@_warning:nxx {defining-font} {#1} {#2}
35 }
36 {
37 <debug>\typeout{Font~ family~ already~ defined.}
38 }
39 \group_end:
40
41 \tl_set_eq:NN \l_fontspec_family_tl \g_@@_nfss_family_tl
42 }

```

(End of definition for \@@_select_font_family:nn. This function is documented on page ??.)

`\fontspec_select:nn` This old name has been used by 3rd party packages so for compatibility:

```

43 \cs_set_eq:NN \fontspec_select:nn \@@_select_font_family:nn %% deprecated, for compatibility o

```

(End of definition for \fontspec_select:nn. This function is documented on page ??.)

`\@@_sanitise_fontname:Nn` Assigns font name #2 to token list variable #1 and strips extension(s) from it in the case of an external font. We strip spaces for luatex for consistency with luaotload, although I'm not sure this is necessary any more. At one stage this also lowercased the name, but this step has been removed unless someone can remind me why it was necessary.

```

44 \cs_new:Nn \@@_sanitise_fontname:Nn
45 {
46   \tl_set:Nx #1 {#2}
47 <LU> \tl_remove_all:Nn #1 {~}
48   \clist_map_inline:Nn \l_@@_extensions_clist
49   {
50     \tl_if_in:NnT #1 {##1}
51     {
52       \tl_remove_once:Nn #1 {##1}
53       \tl_set:Nn \l_@@_extension_tl {##1}
54       \clist_map_break:
55     }
56   }
57 }

```

(End of definition for \@@_sanitise_fontname:Nn. This function is documented on page ??.)

`\@@_if_detect_external:nT` Check if either the fontname ends with a known font extension.

```

58 \prg_new_conditional:Nnn \@@_if_detect_external:n {T}
59 {
60 <debug> \typeout{:: @@_if_detect_external:n { \exp_not:n {#1} } }
61   \clist_map_inline:Nn \l_@@_extensions_clist
62   {

```

```

63     \bool_set_false:N \l_@@_tmpa_bool
64     \exp_args:Nx % <- this should be handled earlier
65     \tl_if_in:nnT {#1 <= end_of_string} {##1 <= end_of_string}
66     { \bool_set_true:N \l_@@_tmpa_bool \clist_map_break: }
67   }
68   \bool_if:NTF \l_@@_tmpa_bool \prg_return_true: \prg_return_false:
69 }

```

(End of definition for \@@_if_detect_external:nT. This function is documented on page ??.)

\@@_init_ttc:n For TTC fonts we assume they will be loading the italic/bold fonts from the same file, so prepopulate the fontnames to avoid needing to do it manually.

```

70 \cs_new:Nn \@@_init_ttc:n
71 {
72   \str_if_eq:eeT { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
73   {
74     \@@_sanitise_fontname:Nn \l_@@_fontname_it_tl {#1}
75     \@@_sanitise_fontname:Nn \l_@@_fontname_bf_tl {#1}
76     \@@_sanitise_fontname:Nn \l_@@_fontname_bfit_tl {#1}
77   }
78 }

```

(End of definition for \@@_init_ttc:n. This function is documented on page ??.)

\@@_load_external_fontoptions:N Load a possible .fontspec font configuration file. This file could set font-specific options for the font about to be loaded. The parameter should be a tokenlist containing a sanitised fontname.

```

79 \cs_new:Nn \@@_load_external_fontoptions:N
80 {
81   \bool_if:NT \l_@@_fontcfg_bool
82   {
83     <debug> \typeout{: : \@@_load_external_fontoptions:N \exp_not:N #1 }
84     \tl_set:Nx \l_@@_ext_filename_tl {#1.fontspec}
85     \tl_remove_all:Nn \l_@@_ext_filename_tl {-}
86     \prop_if_in:NVF \g_@@_fontopts_prop #1
87     {
88       \exp_args:No \file_if_exist:nT { \l_@@_ext_filename_tl }
89       { \file_input:n { \l_@@_ext_filename_tl } }
90     }
91   }
92 }

```

(End of definition for \@@_load_external_fontoptions:N. This function is documented on page ??.)

\@@_extract_all_features:

```

93 \cs_new:Nn \@@_extract_all_features:n
94 {
95   <debug> \typeout{: : \@@_extract_all_features:n { \unexpanded {#1} } }
96   \bool_if:NTF \l_@@_disable_defaults_bool
97   {
98     \clist_set:Nx \l_@@_all_features_clist {#1}
99   }

```

```

100     {
101     \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
102     { \clist_clear:N \l_@@_fontopts_clist }
103
104     \prop_get:NVNF \g_@@_fontopts_prop \l_@@_family_label_tl \l_@@_family_fontopts_clist
105     { \clist_clear:N \l_@@_family_fontopts_clist }
106     \tl_clear:N \l_@@_family_label_tl
107
108     \clist_set:Nx \l_@@_all_features_clist
109     {
110     \g_@@_default_fontopts_clist,
111     \l_@@_family_fontopts_clist,
112     \l_@@_fontopts_clist,
113     #1
114     }
115     }
116 }

```

(End of definition for \@@_extract_all_features:. This function is documented on page ??.)

\@@_preparse_features: #1 : feature options
#2 : font name

Perform the (multi-step) feature parsing process.

Convert the requested features to font definition strings. First the features are parsed for information about font loading (whether it's a named font or external font, etc.), and then information is extracted for the names of the other shape fonts.

```

117 \cs_new:Nn \@@_preparse_features:
118 {
119 (debug) \typeout{: : \@@_preparse_features:}

```

Detect if external fonts are to be used, possibly automatically, and parse fontspec features for bold/italic fonts and their features.

```

120
121 \@@_keys_set_known:nxN {fontspec-preparse-external}
122 { \l_@@_all_features_clist }
123 \l_@@_keys_leftover_clist
124

```

When `\l_fontspec_fontname_tl` is augmented with a prefix or whatever to create the name of the upright font (`\l_@@_fontname_up_tl`), this latter is the new 'general font name' to use.

```

125 \tl_set_eq:NN \l_fontspec_fontname_tl \l_@@_fontname_up_tl
126 \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_keys_leftover_clist}
127 \l_@@_keys_leftover_clist
128 \@@_keys_set_known:nxN {fontspec-preparse} {\l_@@_keys_leftover_clist}
129 \l_@@_fontfeat_clist
130 }

```

(End of definition for \@@_preparse_features:. This function is documented on page ??.)

\@@_load_font:

```

131 \cs_new:Nn \@@_load_font:
132 {

```

```

133 <debug>\typeout{:: @@_load_font}
134
135 <debug>\typeout{Set~ base~ font~ for~ preliminary~ analysis: \@@_construct_font_call:nn { \l_@
136   \@@_primitive_font_set:NnnF \l_@@_test_font
137   { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
138   { \f@size pt - 2sp }
139   { \@@_error:nx {font-not-found} {\l_@@_fontname_up_tl} }
140
141 <debug>\typeout{Set~ base~ font~ properly: \@@_construct_font_call:nn { \l_@@_fontname_up_tl }
142   \@@_set_font_type:N \l_@@_test_font
143   \@@_primitive_font_gset:0nn \l_@@_fontface_cs_tl
144   { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
145   { \f@size pt + 2sp }
146
147   \l_@@_fontface_cs_tl % this is necessary for LuaLaTeX to check the scripts properly
148
149 }

```

(End of definition for \@@_load_font:. This function is documented on page ??.)

\@@_construct_font_call:nn Constructs the complete font invocation. #1 : Base name

#2 : Extension

#3 : TTC Index

#4 : Renderer

#5 : Optical size

#6 : Font features

We check if ** are empty and if so don't add in the separator colon.

```

150 \cs_new:Nn \@@_construct_font_call:nnnnnn
151 {
152 <XE> " \@@_fontname_wrap:n { #1 #2 #3 }
153 <LU> " \@@_fontname_wrap:n { #1 #2 } #3
154   #4 #5
155   \str_if_eq:eeF {#6}{ } {:#6} "
156 }

```

In practice, we don't use the six-argument version, since most arguments are constructed on-the-fly:

```

157 \cs_new:Nn \@@_construct_font_call:nn
158 {
159   \@@_construct_font_call:nnnnnn
160   {#1}
161   \l_@@_extension_tl
162   \l_@@_ttc_index_tl
163   \l_@@_renderer_tl
164   \l_@@_optical_size_tl
165   {#2}
166 }

```

(End of definition for \@@_construct_font_call:nn. This function is documented on page ??.)

`\@@_font_is_file:` The `\@@_fontname_wrap:n` command takes the font name and either passes it through unchanged or wraps it in the syntax for loading a font ‘by filename’. For LuaTeX there are two kinds of filename based loading supported: Regular filename lookups which include system fonts and lookups restricted to kpse.

```

167 \cs_new:Nn \@@_font_is_name:
168   {
169 (XE) \cs_set_eq:NN \@@_fontname_wrap:n \use:n
170 (LU) \cs_set:Npn \@@_fontname_wrap:n ##1 { name: ##1 }
171   }

172 \cs_new:Nn \@@_font_is_file:
173   {
174     \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
175   }

176 (*LU)
177 \cs_new:Nn \@@_font_is_kpse:
178   {
179     \cs_set:Npn \@@_fontname_wrap:n ##1 { kpse: ##1 }
180   }
181 </LU>
182 (XE)\cs_new_eq:NN \@@_font_is_kpse: \@@_font_is_file:

```

(End of definition for \@@_font_is_file:, \@@_font_is_name:, and \@@_font_is_kpse:. These functions are documented on page ??.)

`\@@_set_scriptlang:` Only necessary for OpenType fonts. First check if the font supports scripts, then apply defaults if none are explicitly requested. Similarly with the language settings.

```

183 \cs_new:Nn \@@_set_scriptlang:
184   {
185 (debug) \typeout{:: _set_scriptlang:}
186     \bool_if:NT \l_@@_firsttime_bool
187       {
188         \tl_if_empty:NF \l_@@_script_name_tl
189           {
190 (debug) \typeout{::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
191           \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
192           \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
193         }
194       }
195   }

```

(End of definition for \@@_set_scriptlang:. This function is documented on page ??.)

`\@@_get_features:Nn` This macro is a wrapper for `\keys_set:n` which expands and adds a default specification to the original passed options. It begins by initialising the commands used to hold font-feature specific strings. Its argument is any additional features to prepend to the default.

Do not set the colour if not explicitly spec'd else `\color` (using specials) will not work.

```

196 \cs_new:Nn \@@_get_features:n
197   {
198 (debug) \typeout{:: \@@_get_features:Nn { \exp_not:n {##1} } }
199     \@@_init_fontface:
200     \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,##1}

```

```

201     \l_@@_keys_leftover_clist
202     \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
203 (*XE)
204     \bool_if:NTF \l_@@_ot_bool
205     {
206 (debug) \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
207     \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
208     }
209     {
210 (debug) \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features:~"\l_@@_keys_leftover_c
211     \bool_if:nT { \l_@@_atsui_bool || \l_@@_graphite_bool }
212     { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
213     }
214 (/XE)
215 (*LU)
216 (debug) \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
217     \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
218 (/LU)
219
220     \tl_if_empty:NF \l_@@_mapping_tl
221     { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
222
223     \str_if_eq:eeF { \l_@@_hexcol_tl \l_@@_opacity_tl }
224     { \c_@@_hexcol_tl \c_@@_opacity_tl }
225 (XE)     { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
226 (LU)     { \@@_update_featstr:n { color = {\l_@@_hexcol_tl\l_@@_opacity_tl} } }
227     }

```

(End of definition for \@@_get_features:Nn. This function is documented on page ??.)

\@@_save_family_needed:nTF Check if the family is unique and, if so, save its information. (\addfontfeature and other macros use this data.) Then the font family and its shapes are defined in the NFSS.

Now we have a unique (in fact, too unique!) string that contains the family name and every option in abbreviated form. This is used with a counter to create a simple NFSS family name for the font we're selecting.

```

228 \prg_new_conditional:Nnn \@@_save_family_needed:n { TF }
229 {
230
231 (debug) \typeout{save~ family:~ #1}
232 (debug) \typeout{== fontid_tl: "\l_@@_fontid_tl".}
233
234 \tl_if_empty:NTF \l_@@_nfss_fam_tl
235 {
236     \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
237     {
238         \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
239         \prg_return_false:
240     }
241     {
242         \tl_set:Nx \l_@@_tmp_tl {#1}
243         \tl_remove_all:Nn \l_@@_tmp_tl { ~ }

```



```

244     \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl
245     \prg_return_true:
246   }
247 }
248 {
249   \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
250   \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl _prop }
251   \prg_return_true:
252 }
253 }
254 \cs_new:Nn \@@_save_fontid_family:nn
255 {
256   \prop_get:NnNTF \g_@@_family_int_prop {#2} \l_@@_tmp_tl
257   {
258     \tl_set:Nx \l_@@_tmp_tl
259     { \int_eval:n { \l_@@_tmp_tl + 1 } }
260   }
261   { \tl_set:Nn \l_@@_tmp_tl { 0 } }
262   \prop_gput:NnV \g_@@_family_int_prop {#2} \l_@@_tmp_tl
263   \tl_gset:Nx \g_@@_nfss_family_tl { #2 ( \l_@@_tmp_tl ) }
264   \prop_gput:NnV \g_@@_fontid_family_prop {#1} \g_@@_nfss_family_tl
265 }
266 \cs_generate_variant:Nn \@@_save_fontid_family:nn { VV }

```

(End of definition for \@@_save_family_needed:nTF. This function is documented on page ??.)

\@@_save_family:nn Saves the relevant font information for future processing.

```

267 \cs_new:Nn \@@_save_family:nn
268 {
269   \@@_save_fontinfo:n {#2}
270   \@@_find_autofonts:
271   \DeclareFontFamily{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{}
272   \@@_set_faces:
273   \@@_info:nxx {defining-font} {#1} {#2}
274 }

```

(End of definition for \@@_save_family:nn. This function is documented on page ??.)

\@@_save_fontinfo:n Saves the relevant font information for future processing.

```

275 \cs_new:Nn \@@_save_fontinfo:n
276 {
277   \prop_new:c {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop}
278   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
279   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options} { \l_@@_all_features }
280   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
281   {
282     \@@_construct_font_call:nn {\l_fontspeg_fontname_tl}
283     { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist \@@_get_variations: }
284   }
285   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
286   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num} \l_@@_language_int
287   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_@@_script_tl

```

```

288     \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_@@_lang_tl
289   }

```

(End of definition for \@@_save_fontinfo:n. This function is documented on page ??.)

1.2 Setting font shapes in a family

All NFSS specifications take their default values, so if any of them are redefined, the shapes will be selected to fit in with the current state. For example, if `\bfdefault` is redefined to `b`, all bold shapes defined by this package will also be assigned to `b`.

The combination shapes are searched first because they use information that may be redefined in the single cases. E.g., if no bold font is specified then `set_autofont` will attempt to set it. This has subtle/small ramifications on the logic of choosing the bold italic font.

`\@@_find_autofonts:`

```

290 \cs_new:Nn \@@_find_autofonts:
291   {
292     \bool_if:NF {\l_@@_noit_bool || \l_@@_nobf_bool}
293     {
294       \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
295       \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
296       \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspeg_fontname_tl} {/BI}
297     }
298
299     \bool_if:NF \l_@@_nobf_bool
300     {
301       \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspeg_fontname_tl} {/B}
302     }
303
304     \bool_if:NF \l_@@_noit_bool
305     {
306       \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspeg_fontname_tl} {/I}
307     }
308
309     \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
310   }

```

(End of definition for \@@_find_autofonts:. This function is documented on page ??.)

`\@@_set_faces:`

```

311 \cs_new:Nn \@@_set_faces:
312   {
313     \@@_add_nfssfont:nmmm \mddefault \shapedefault \l_fontspeg_fontname_tl \l_@@_fontfeat_up_c
314     \@@_add_nfssfont:nmmm \bfdefault \shapedefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_c
315     \@@_add_nfssfont:nmmm \mddefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_c
316     \@@_add_nfssfont:nmmm \mddefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_c
317     \@@_add_nfssfont:nmmm \mddefault \swdefault \l_@@_fontname_sw_tl \l_@@_fontfeat_sw_c
318     \@@_add_nfssfont:nmmm \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit
319     \@@_add_nfssfont:nmmm \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl
320     \@@_add_nfssfont:nmmm \bfdefault \swdefault \l_@@_fontname_bfsw_tl \l_@@_fontfeat_bfsw
321     \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nmmnnn ##2 }
322   }

```

```

323 \cs_new:Nn \@@_set_faces_aux:nnnnn
324 {
325   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
326   \@@_make_font_shapes:Nnnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
327 }

```

(End of definition for \@@_set_faces:. This function is documented on page ??.)

`\fontspec_complete_fontname:Nn` This macro defines #1 as the input with any * tokens of its input replaced by the font name. This lets us define supplementary fonts in full (“Baskerville Semibold”) or in abbreviation (“* Semibold”).

```

328 \cs_new:Nn \fontspec_complete_fontname:Nn
329 {
330   \tl_set:Nx #1 {#2}
331   \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
332   \tl_remove_all:Nn #1 {~}
333 }

```

(End of definition for \fontspec_complete_fontname:Nn. This function is documented on page ??.)

`\@@_add_nfssfont:nnnn` #1 : series
#2 : shape
#3 : fontname
#4 : fontspec features

```

334 \cs_new:Nn \@@_add_nfssfont:nnnn
335 {
336   \tl_set:Nx \l_@@_this_font_tl {#3}
337
338   \tl_if_empty:xTF {#4}
339   { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
340   { \@@_keys_set_known:nxN {fontspec-preparse-nested} {#4} \l_@@_tmp_tl }
341
342   \tl_if_empty:NF \l_@@_this_font_tl
343   {
344     \prop_put:Nxx \l_@@_nfssfont_prop {#1/#2}
345     { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
346   }
347 }

```

(End of definition for \@@_add_nfssfont:nnnn. This function is documented on page ??.)

1.2.1 Fonts

`\@@_set_font_type:N` Now check if the font is to be rendered with `ATSU1` or `Harfbuzz`. This will either be automatic (based on the font type), or specified by the user via a font feature.

This macro sets booleans accordingly depending if the font in `\l_fontspec_test_font` is an AAT font or an OpenType font or a font with feature axes (either AAT or Multiple Master), respectively.

```

348 \cs_new:Nn \@@_set_font_type:N
349 {
350   \debug \typeout{: : @@_set_font_type:}

```

```

351 ⟨*XE⟩
352   \bool_set_false:N \l_@@_tfm_bool
353   \bool_set_false:N \l_@@_atsui_bool
354   \bool_set_false:N \l_@@_ot_bool
355   \bool_set_false:N \l_@@_mm_bool
356   \bool_set_false:N \l_@@_graphite_bool
357   \ifcase\XeTeXfonttype #1
358 ⟨debug⟩ \typeout{::: TFM}
359   \bool_set_true:N \l_@@_tfm_bool
360   \or
361 ⟨debug⟩ \typeout{::: AAT}
362   \bool_set_true:N \l_@@_atsui_bool
363   \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/AAT} }
364   \ifnum\XeTeXcountvariations #1 > 0\relax
365 ⟨debug⟩ \typeout{::: MM}
366   \bool_set_true:N \l_@@_mm_bool
367   \fi
368   \or
369 ⟨debug⟩ \typeout{::: OpenType}
370   \bool_set_true:N \l_@@_ot_bool
371   \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/OT} }
372   \or
373 ⟨debug⟩ \typeout{::: Graphite}
374   \bool_set_true:N \l_@@_graphite_bool
375   \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/GR} }
376   \fi
377 ⟨/XE⟩

```

If automatic, the `\l_@@_renderer_tl` token list will still be empty (other suffices that could be added will be later in the feature processing), and if it is indeed still empty, assign it a value so that the other weights of the font are specifically loaded with the same renderer.

LuaTeX only supports one:

```

378 ⟨*LU⟩
379   \bool_set_true:N \l_@@_ot_bool
380 ⟨/LU⟩
381   }

```

(End of definition for `\@@_set_font_type:N`. This function is documented on page ??.)

```

\@@_set_autofont:Nnn #1 : Font name tl
                    #2 : Base font name
                    #3 : Font name modifier

```

This function looks for font with `⟨name⟩` and `⟨modifier⟩ #2#3`, and if found (i.e., different to font with name #2) stores it in tl #1. A modifier is something like `/B` to look for a bold font, for example.

We can't match external fonts in this way (in X_ƎTeX anyway; todo: test with LuaTeX). If `⟨font name tl⟩` is not empty, then it's already been specified by the user so abort. If `⟨Base font name⟩` is not given, we also abort for obvious reasons.

If `⟨font name tl⟩` is empty, then proceed. If not found, `⟨font name tl⟩` remains empty. Otherwise, we have a match.

```

382 \cs_new:Nn \@@_set_autofont:Nnn

```

```

383 {
384   \bool_if:NF \l_@@_external_bool
385   {
386     \tl_if_empty:xF {#2}
387     {
388       \tl_if_empty:NT #1
389       {
390         \@@_if_autofont:nnTF {#2} {#3}
391         { \tl_set:Nx #1 {#2#3} }
392         { \@@_info:nx {no-font-shape} {#2#3} }
393       }
394     }
395   }
396 }
397 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
398 {
399   \group_begin:
400   \@@_primitive_font_set:Nnn \l_@@_tmpa_font { \@@_construct_font_call:nn {#1} { \l_@@_pre
401   \@@_primitive_font_set:Nnn \l_@@_tmpb_font { \@@_construct_font_call:nn {#1#2} { \l_@@_pre
402   \cs_if_eq:NNTF \l_@@_tmpa_font \l_@@_tmpb_font
403   { \group_end: \prg_return_false: }
404   { \group_end: \prg_return_true: }
405 }

```

(End of definition for \@@_set_autofont:Nnn. This function is documented on page ??.)

\@@_make_font_shapes:Nnnnn #1 : Font name
#2 : Font series
#3 : Font shape
#4 : Font features
#5 : Size features

This macro eventually uses `\DeclareFontShape` to define the font shape in question.

```

406 \cs_new:Nn \@@_make_font_shapes:Nnnnn
407 {
408   \group_begin:
409   \@@_keys_set_known:nxN {fontspec-prepare-external} { #4 } \l_@@_leftover_clist
410   \@@_load_fontname:Nn \l_fontspeg_fontname_tl {#1}
411   \@@_declare_shape:nxxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
412   \group_end:
413 }
414 \cs_new:Nn \@@_load_fontname:Nn
415 {
416 (debug) \typeout{: @@@_load_fontname:Nn \exp_not:N #1 (#1) {#2} }
417 \@@_sanitise_fontname:Nn #1 {#2}
418 \@@_load_external_fontoptions:N #1
419 \prop_get:NVNF \g_@@_fontopts_prop #1 \l_@@_fontopts_clist
420 { \clist_clear:N \l_@@_fontopts_clist }
421 \keys_set_groups:nnV {fontspec/fontname} {getfontname} \l_@@_fontopts_clist
422 \@@_primitive_font_set:OnnF \l_@@_fontface_cs_tl
423 { \@@_construct_font_call:nn {#1} { \l_@@_pre_feat_sclist } } { \f@size pt + 2sp }
424 { \@@_error:nx {font-not-found} {#2} }

```

```

425 }
426 \keys_define:nn {fontspec/fontname}
427 {
428   Font .tl_set:N = \l_fontspec_fontname_tl ,
429   Font .groups:n = {getfontname} ,
430 }

```

(End of definition for \@@_make_font_shapes:Nnnnn. This function is documented on page ??.)

```

\@@_declare_shape:nmmm #1 : Font series
                       #2 : Font shape
                       #3 : Font features
                       #4 : Size features

```

Wrapper for \DeclareFontShape. And finally the actual font shape declaration using \l_@@_nfss_tl defined above. \l_@@_postadjust_tl is defined in various places to deal with things like the hyphenation character and interword spacing.

The main part is to loop through SizeFeatures arguments, which are of the form
 SizeFeatures={{<one>},{<two>},{<three>}}.

```

431 \cs_new:Nn \@@_declare_shape:nmmm
432 {
433 (debug)\typeout{~ declare_shape:~{\l_fontspec_fontname_tl}~{#1}~{#2}}
434   \tl_build_begin:N \l_@@_nfss_tl
435   \tl_build_begin:N \l_@@_nfss_sc_tl
436   \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspec_fontname_tl
437
438   \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
439
440   \tl_build_end:N \l_@@_nfss_tl
441   \tl_build_end:N \l_@@_nfss_sc_tl
442
443   \@@_declare_shapes_normal:nn {#1} {#2}
444   \@@_declare_shapes_smcaps:nn {#1} {#2}
445   \@@_declare_shape_slanted:nn {#1} {#2}
446   \@@_declare_shapes_bx:nn {#1} {#2}
447   \@@_declare_shape_loginfo:nn {#1} {#2}
448 }
449 \cs_generate_variant:Nn \@@_declare_shape:nmmm {nxxx}

```

(End of definition for \@@_declare_shape:nmmm. This function is documented on page ??.)

```

\@@_setup_single_size:nn

```

```

450 \cs_new:Nn \@@_setup_single_size:nn
451 {
452   \tl_clear:N \l_@@_size_tl
453   \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
454
455   \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
456   \l_@@_sizing_leftover_clist
457   \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
458 (debug)\typeout{== size:~\l_@@_size_tl}

```

```

459 % "normal"
460
461 \@@_load_fontname:Nn \l_fontspec_fontname_tl {\l_@@_sizedfont_tl}
462 \@@_setup_nfss:Nnnn \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
463 (debug) \typeout{===~ sized~ font:~ \l_@@_sizedfont_tl}
464
465 % small caps
466 \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist
467
468 \bool_if:NF \l_@@_nosc_bool
469 {
470   \tl_if_empty:NTF \l_@@_fontname_sc_tl
471   {
472     \@@_make_smallcaps:TF
473     {
474 (debug)\typeout{====~Small~ caps~ found.}
475       \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
476     }
477   }
478 (debug)\typeout{====~Small~ caps~ not~ found.}
479   \bool_set_true:N \l_@@_nosc_bool
480 }
481 }
482 { \@@_load_fontname:Nn \l_fontspec_fontname_tl {\l_@@_fontname_sc_tl} }% local for e
483 }
484
485 \bool_if:NF \l_@@_nosc_bool
486 {
487   \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
488   {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
489 }
490 }

```

(End of definition for \@@_setup_single_size:n. This function is documented on page ??.)

\@@_setup_nfss:Nnnn

```

491 \cs_new:Nn \@@_setup_nfss:Nnnn
492 {
493 (debug)\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}
494
495   \@@_get_features:n { #2 , #3 , #4 }
496 (debug)\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist \@@_get_variations:}
497
498   \tl_if_empty:NF \l_@@_scale_tl
499   {
500     \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
501   }
502
503   \tl_build_put_right:Nx #1
504   {
505     <\l_@@_size_tl> \l_@@_scale_tl
506     \@@_construct_font_call:n { \l_fontspec_fontname_tl }

```

```

507         { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist \@@_get_variations: }
508     }
509 }

```

(End of definition for \@@_setup_nfss:Nnnn. This function is documented on page ??.)

`\@@_declare_shapes_normal:nn`

```

510 \cs_new:Nn \@@_declare_shapes_normal:nn
511 {
512     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
513     {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
514 }

```

(End of definition for \@@_declare_shapes_normal:nn. This function is documented on page ??.)

`\@@_declare_shapes_smcaps:nn`

```

515 \cs_new:Nn \@@_declare_shapes_smcaps:nn
516 {
517     \tl_if_empty:NF \l_@@_nfss_sc_tl
518     {
519         \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl} {#1}
520         { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
521     }
522 }
523 \cs_new:Nn \@@_combo_sc_shape:n
524 {
525     \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
526     { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
527     { \scdefault#1 }
528 }

```

(End of definition for \@@_declare_shapes_smcaps:nn. This function is documented on page ??.)

`\@@_DeclareFontShape:nnnnnn`

```

529 \cs_new:Nn \@@_DeclareFontShape:nnnnnn
530 {
531     <debug>\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
532     \group_begin:
533     \normalsize
534     \cs_undefine:c {#1/#2/#3/#4/\f@size}
535     \group_end:
536     \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
537 }
538 \cs_generate_variant:Nn \@@_DeclareFontShape:nnnnnn {xxxxxx}

```

`\@@_declare_shape_slanted:nn`

This extra stuff for the slanted shape substitution is a little bit awkward. We define the slanted shape to be a synonym for it when (a) we're defining an italic font, but also (b) when the default slanted shape isn't 'it'. (Presumably this turned up once in a test and I realised it caused problems. I doubt this would happen much.)

We should test when a slanted font has been specified and not run this code if so, but the `\@@_set_slanted:` code will overwrite this anyway if necessary.


```

539 \cs_new:Nn \@@_declare_shape_slanted:nn
540 {
541   \bool_if:nT
542   {
543     \str_if_eq_p:ee {#2} {\itdefault} &&
544     !(\str_if_eq_p:ee {\itdefault} {\sldefault})
545   }
546   {
547     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}
548     {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
549   }
550 }

```

Similar processing for setting up b/bx substitutions.

```

\@@_declare_shapes_bx:nn 551 \cs_new:Nn \@@_declare_shapes_bx:nn
552 {
553   \bool_if:nT
554   {
555     \str_if_eq_p:ee {#1} {\bfdefault} &&
556     !(\str_if_eq_p:ee {\bfdefault} {bx})
557   }
558   {
559     % bx/?
560     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
561     {bx} {#2}
562     { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
563     { \l_@@_postadjust_tl }
564
565     % bx/sc -> b/sc
566     \tl_if_empty:NF \l_@@_nfss_sc_tl
567     {
568       \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
569       {bx} { \@@_combo_sc_shape:n {#2} }
570       { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
571       { \l_@@_postadjust_tl }
572     }
573
574     % bx/sl -> bx/it
575     \bool_if:nT
576     {
577       \str_if_eq_p:ee {#2} {\itdefault} &&
578       !(\str_if_eq_p:ee {\itdefault} {\sldefault})
579     }
580     {
581       \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
582       {bx} {\sldefault}
583       { <->ssub*\g_@@_nfss_family_tl/bx/\itdefault }
584       { \l_@@_postadjust_tl }
585     }
586   }
587 }
588 }

```

Lastly some informative messaging.

```

\l_@@_declare_shape_loginfo:nn 589 \cs_new:Nn \l_@@_declare_shape_loginfo:nn
590 {
591   \tl_gput_right:Nx \g_@@_defined_shapes_tl
592   {
593     \exp_not:n { \ }
594     -- \exp_not:N \str_case:nn {#1/#2}
595     {
596       {\mddefault/\shapedefault} {'normal'~}
597       {\bfdefault/\shapedefault} {'bold'~}
598       {\mddefault/\itdefault} {'italic'~}
599       {\mddefault/\sldefault} {'slanted'~}
600       {\mddefault/\swdefault} {'swash'~}
601       {\bfdefault/\itdefault} {'bold~ italic'~}
602       {\bfdefault/\sldefault} {'bold~ slanted'~}
603       {\bfdefault/\swdefault} {'bold~ swash'~}
604     } (#1/#2)~
605     with~ NFSS~ spec.:~
606     \l_@@_nfss_tl
607     \exp_not:n { \ }
608     -- \exp_not:N \str_case:nn { #1 / \l_@@_combo_sc_shape:n {#2} }
609     {
610       {\mddefault/\scdefault} {'small~ caps'~}
611       {\bfdefault/\scdefault} {'bold~ small~ caps'~}
612       {\mddefault/\scitdefault} {'italic~ small~ caps'~}
613       {\bfdefault/\scitdefault} {'bold~ italic~ small~ caps'~}
614       {\mddefault/\scsldefault} {'slanted~ small~ caps'~}
615       {\bfdefault/\scsldefault} {'bold~ slanted~ small~ caps'~}
616     }~( #1 / \l_@@_combo_sc_shape:n {#2} )~
617     with~ NFSS~ spec.:~
618     \l_@@_nfss_sc_tl
619     \tl_if_empty:xF {\l_@@_postadjust_tl}
620     {
621       \exp_not:N \ and~ font~ adjustment~ code:
622       \exp_not:N \ \l_@@_postadjust_tl
623     }
624   }
625 }

```

Maybe `\str_if_eq:eeF` would be better?

1.2.2 Features

These are the features always applied to a font selection before other features.

```

\l_@@_pre_feat_sclist 626 \tl_set:Nn \l_@@_pre_feat_sclist
627 (*XE)
628 {
629   \bool_if:NT \l_@@_ot_bool
630   {
631     \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }
632     \tl_if_empty:NF \l_@@_lang_tl { language = \l_@@_lang_tl ; }
633   }

```

```

634 }
635 </XE>
636 (*LU)
637 {
638     mode = \l_@@_mode_tl ;
639     \tl_if_empty:NF \l_@@_shaper_tl { shaper = \l_@@_shaper_tl ; }
640     \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }
641     \tl_if_empty:NF \l_@@_lang_tl { language = \l_@@_lang_tl ; }
642 }
643 </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF 644 <LU>\cs_new:Nn \@@_make_smallcaps:TF
645 <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
646 {
647     \exp_args:No \@@_check_ot_feat:NnTF \l_@@_fontface_cs_tl {smcp} {#1} {#2}
648 }
649 (*XE)
650 \cs_new:Nn \@@_make_smallcaps:TF
651 {
652     \bool_if:NTF \l_@@_ot_bool
653     { \@@_make_ot_smallcaps:TF {#1} {#2} }
654     {
655         \bool_if:NT \l_@@_atsui_bool
656         {
657             \exp_args:No \@@_make_AAT_feature_string:NnnTF
658             \l_@@_fontface_cs_tl {3} {3} {#1} {#2}
659         }
660     }
661 }
662 </XE>

```

\g_@@_rawfeatures_sclist is the string used to define the list of specific font features. Each time another font feature is requested, this macro is used to add that feature to the list. Font features are separated by semicolons.

```

\@@_update_featstr:n 663 \cs_new:Nn \@@_update_featstr:n
664 {
665     <debug> \typeout{::: @@@_update_featstr:n {#1}}
666     \bool_if:NF \l_@@_firsttime_bool
667     {
668         \tl_gset:Nx \g_@@_single_feat_tl { #1 }
669     <debug> \typeout{:::~ Adding~ feature.}
670     \tl_gput_right:Nx \g_@@_rawfeatures_sclist {#1;}
671     }
672 }

```

```

\@@_remove_clashing_featstr:n 673 \cs_new:Nn \@@_remove_clashing_featstr:n
674 {
675     <debug> \typeout{::: @@@_remove_clashing_featstr:n {#1}}
676     \clist_map_inline:nn {#1}

```

```

677     {
678 (debug)       \typeout{:::~ Removing~ feature~ "##1;}
679     \tl_gremove_all:Nn \g_@@_rawfeatures_sclist {##1;}
680     }
681   }
682 \cs_generate_variant:Nn \@@_remove_clashing_featstr:n {x}

```

\@@_get_variations: builds the feature string representing the current variation instance

\@@_get_variations: and/or axis settings.

```

683 \cs_generate_variant:Nn \tl_tail:n { e }
684 \cs_new:Nn \@@_format_axis:nn
685   {
686     , #1 = #2
687   }
688 \cs_new:Nn \@@_get_variations:
689   {
690     \tl_if_empty:NF \g_@@_instance_tl
691     {
692       instance = { \g_@@_instance_tl };
693     }
694     \prop_if_empty:NF \g_@@_rawvariations_prop
695     {
696       axis = {
697         \tl_tail:e {
698           \prop_map_function:NN \g_@@_rawvariations_prop \@@_format_axis:nn
699         }
700       };
701     }
702   }

```

1.3 Initialisation

Initialisations that need to occur once per fontspec font invocation. (Some of these may be

\@@_init: redundant. Check whether they're assigned to globally or not.)

```

703 \cs_set:Npn \@@_init:
704   {
705 (debug) \typeout{:: @@@_init:}
706     \bool_set_false:N \l_@@_ot_bool
707     \bool_set_true:N \l_@@_firsttime_bool
708     \@@_font_is_name:
709     \tl_clear:N \l_@@_font_path_tl
710     \tl_clear:N \l_@@_optical_size_tl
711     \tl_clear:N \l_@@_ttc_index_tl
712     \tl_clear:N \l_@@_renderer_tl
713     \tl_gclear:N \g_@@_defined_shapes_tl
714     \tl_gclear:N \g_@@_curr_series_tl
715     \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspec_encoding_tl
716 (*LU)
717     \tl_set:Nn \l_@@_mode_tl {node}
718     \int_set:Nn \prehyphenchar { \- } % fixme
719     \int_zero:N \posthyphenchar % fixme

```

```

720 \int_zero:N \preexhyphenchar % fixe
721 \int_zero:N \postexhyphenchar % fixe
722 </LU>
723 }

```

Executed in \@@_get_features:Nn.

```

\@@_init_fontface: 724 \cs_new:Nn \@@_init_fontface:
725 {
726 \tl_gclear:N \g_@@_rawfeatures_sclist
727 \prop_gclear:N \g_@@_rawvariations_prop
728 \tl_gclear:N \g_@@_instance_tl
729 \tl_clear:N \l_@@_scale_tl
730 \tl_set_eq:NN \l_@@_opacity_tl \c_@@_opacity_tl
731 \tl_set_eq:NN \l_@@_hexcol_tl \c_@@_hexcol_tl
732 \tl_set_eq:NN \l_@@_postadjust_tl \c_@@_postadjust_tl
733 \tl_clear:N \l_@@_wordspace_adjust_tl
734 \tl_clear:N \l_@@_punctspace_adjust_tl
735 }

```

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

```

\@@_ot_validate_tag:n 736 (*LU)
737 \cs_new_protected:Nn \@@_ot_validate_tag:n
738 {
739 \@@_ot_validate_tag:w #1 \q_nil
740 }
741 \cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
742 \cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
743 {
744 \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
745 { \@@_ot_validate_tag_aux:w #2 \c_empty_tl \c_empty_tl \q_nil }
746 { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
747 }
748 \cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
749 {
750 \int_compare:nT { \tl_count:n {#5} > 2 }
751 { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
752 }
753 </LU>

```

\@@_iv_str_to_num:Nn This macro takes a four character string and converts it to the numerical representation required for X_YTeX OpenType script/language/feature purposes. The output is stored in #1.

This code is not used in LuaTeX, as the checking for that engine is done via Lua code provided by luaotfload.

```

754 (*XE)

```

```

755 \cs_new:Nn \@@_iv_str_to_num:Nn
756   {
757   (debug)\typeout{\_iv_str_to_num:~#1~/~#2}
758     \@@_strip_leading_sign:Nw #1#2 \q_nil
759   }
760 \cs_generate_variant:Nn \@@_iv_str_to_num:Nn {Nx}

```

The input can be of the form of any of these: 'abcd', 'abc', 'abc ', 'ab', 'ab ', etc. (It is assumed the first two chars are *always* not spaces.) So this macro reads in the string padded with `\empty s`, and anything beyond four chars is snipped. The `\empty s` then are used to reconstruct the spaces in the string to number calculation.

For backwards compatibility this code also strips a leading + or -.

```

761 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
762   {
763     \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
764       { \@@_iv_str_to_num:w #1 \q_nil #3   \c_empty_tl \c_empty_tl \q_nil }
765       { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
766   }

```

If input string (after sign is stripped) is more than 4 chars, #6 will contain '`excess`' `\c_empty_tl`. Therefore use #6 to verify string length.

```

767 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
768   {
769     \int_compare:nT { \tl_count:n {#6} > 2 }
770       { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }
771
772     \int_set:Nn #1
773       {
774         `#2 * "10000000
775         + `#3 * "100000
776         + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
777         + \ifx \c_empty_tl #5 32 \else `#5 \fi
778       }
779   }
780 \</math>

```

File XI

fontspec-code-opentype.dtx

1 OpenType definitions code

```
\@@_define_opentype_variation_axis:nn 1 \cs_new:Nn \@@_define_opentype_variation_axis:nn
2 {
3   \keys_define:nn {fontspec-opentype}
4     {
5       #1 .code:n = {
6         \prop_gput:Nnn \g_@@_rawvariations_prop { #2 } { ##1 }
7       },
8       #1 .value_required:n = true,
9       #1 .groups:n = {opentype},
10    }
11 }

\@@_define_opentype_feature_group:n 12 \cs_new:Nn \@@_define_opentype_feature_group:n
13 {
14   \keys_define:nn {fontspec-opentype} { #1 .multichoice: , .groups:n = {opentype} }
15 }

#1 : Feature key
\@@_define_opentype_feature:nmnn 16 \cs_new:Nn \@@_feat_prop_add:nn
#2 : Feature option val
17 {
#3 : Check feature — leave empty for no check
18   \tl_if_empty:nF {#1}
#4 : Exact tag string to activate — leave empty for disable only
19   {
#5 : Tags to remove (clist)
20     \prop_if_in:NnF \g_@@_OT_features_prop {#1}
21     {
22       \prop_gput:Nnn \g_@@_OT_features_prop {#1} {#2}
23     }
24   }
25 }
26 \cs_new:Nn \@@_define_opentype_feature:nmnn
27 {
28   \@@_feat_prop_add:nn {#3} {#1\,=#, #2}
29   \tl_if_empty:nTF {#4}
30   {
31     \keys_define:nn {fontspec-opentype}
32     {
33       #1/#2 .code:n =
34       { \@@_remove_clashing_featstr:n {#5} } ,
```

```

35         #1/#2 .groups:n = {opentype}
36     }
37 }
38 {
39     \keys_define:nn {fontspec-opentype}
40     {
41         #1/#2 .code:n =
42         {
43 (debug)     \typeout{::::::::::fontspec-opentype-#1/#2-~#3/#4/#5}
44             \@@_make_OT_feature:nnn {#3} {#4} {#5}
45         } ,
46         #1/#2 .groups:n = {opentype}
47     }
48 }
49 }

```

#1 : Feature key
#2 : Feature option val
#3 : Check feature
#4 : Tag prefix to activate: +#4 = on, -#4 = off.
#5 : Tags to remove in the on case (clist)

\@@_define_opentype_onoffreset:nmnnn

```

50 \cs_new:Nn \@@_feat_off:n {#10ff}
51 \cs_new:Nn \@@_feat_reset:n {#1Reset}
52 \cs_new:Nn \@@_define_opentype_onoffreset:nmnnn
53 {
54     \exp_args:Nnx \@@_define_opentype_feature:nmnnnn {#1} {#2} {#3} {+#4} {#5}
55     \exp_args:Nnx \@@_define_opentype_feature:nmnnnn {#1} { \@@_feat_off:n {#2} } {#3} {-#4}
56     \exp_args:Nnx \@@_define_opentype_feature:nmnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {+#4}
57 }

```

#1 : Feature key
#2 : Feature option val
#3 : Check feature
#4 : Exact tag string to activate
#5 : Tags to remove (clist)

\@@_define_opentype_onreset:nmnnn

```

58 \cs_new:Nn \@@_define_opentype_onreset:nmnnn
59 {
60     \exp_args:Nnx \@@_define_opentype_feature:nmnnnn {#1} {#2} {#3} {#4} {#5}
61     \exp_args:Nnx \@@_define_opentype_feature:nmnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {#4}
62 }

```

1.1 Adding features when loading fonts

When remove clashing features,

1. remove the feature being added (to avoid duplicates);
2. remove the inverse of the feature (to avoid cancellation);
3. finally remove all clashing features.


```

63 \cs_new:Nn \@@_make_OT_feature:nnn
64 {
65 (debug) \typeout{:: \@@_make_OT_feature:nnn \exp_not:n { {#1}{#2}{#3} } }
66
67 \bool_set_true:N \l_@@_proceed_bool
68
69 \tl_if_empty:nF {#1}
70 {
71 \exp_args:No \@@_check_ot_feat:NnF \l_@@_fontface_cs_tl {#1}
72 {
73 \@@_warning:nx {icu-feature-not-exist-in-font} {#1}
74 \bool_set_false:N \l_@@_proceed_bool
75 }
76 }
77
78 \@@_remove_clashing_featstr:x { #2 , \@@_swap_plus_minus:n {#2} , #3 }
79
80 \bool_if:NT \l_@@_proceed_bool { \@@_update_featstr:n {#2} }
81 }
82 \cs_generate_variant:Nn \@@_make_OT_feature:nnn {xxx}
83 \cs_new:Nn \@@_swap_plus_minus:n { \@@_swap_plus_minus_aux:Nq #1 \q_nil }
84 \cs_new:Npn \@@_swap_plus_minus_aux:Nq #1#2 \q_nil
85 { \str_case:nn {#1} { {+} {-#2} {-} {+#2} } }

```

(End of definition for \@@_DeclareFontShape:nnnnn and others. These functions are documented on page ??.)

\@@_check_script:NnTF This macro takes an OpenType script tag and checks if it exists in the current font. \l_@@_script_int is used to store the number corresponding to the script tag string.

```

86 \prg_new_conditional:Nnn \@@_check_script:Nn {TF,T}
87 {
88 (debug)\typeout{:: \_check_script:Nn~#1~/~#2}
89 \bool_if:NnTF \l_@@_never_check_bool
90 { \prg_return_true: }
91 {
92 \bool_if:nTF { \tl_if_empty_p:e {#2} }
93 { \prg_return_false: }
94 {
95 (*XE)
96 (debug)\typeout{::::~~ checking~ script~ #2}
97 \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
98 \int_set:Nn \l_tmpb_int { \XeTeXOTcountscripts #1 }
99 \int_zero:N \l_tmpa_int
100 \bool_set_false:N \l__fontspec_check_bool
101 \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
102 {
103 \ifnum \XeTeXOTscripttag #1 \l_tmpa_int = \l_@@_strnum_int
104 \bool_set_true:N \l__fontspec_check_bool
105 \int_set:Nn \l_tmpa_int { \l_tmpb_int }
106 }
107 \else
108 \int_incr:N \l_tmpa_int
109 \fi

```

```

109     }
110     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
111 </XE>
112 (*LU)
113     \@@_ot_validate_tag:x {#2}
114     \cs_if_eq:NNTF #1 \font
115       { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
116       { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
117 <debug>\typeout{:::~ checking:~"\l_@@_tmp_tl",~ "#2"}
118     \lua_now:e { fontspec.check_ot_script("\l_@@_tmp_tl", "#2") }
119     \bool_if:NTF \l__fontspec_check_bool
120     {
121 <debug>\typeout{:::~ TRUE}
122       \prg_return_true:
123     }
124     {
125 <debug>\typeout{:::~ FALSE}
126       \prg_return_false:
127     }
128 </LU>
129   }
130 }
131 }

```

(End of definition for \@@_check_script:NnTF. This function is documented on page ??.)

\@@_check_lang:NnnTF This macro takes an OpenType language tag and checks if it exists in the current font/script.
 \@@_check_lang:NnTF \l_@@_language_int is used to store the number corresponding to the language tag string.
 The script used is whatever's held in \l_@@_script_int. By default, that's the number corresponding to 'latn'.

```

132 \prg_new_conditional:Nnn \@@_check_lang:Nn {TF}
133 {
134   \@@_check_lang:NnnTF #1 {#2} {\l_@@_script_tl} {\prg_return_true:} {\prg_return_false:}
135 }
136 \prg_new_conditional:Nnn \@@_check_lang:Nnn {TF}
137 {
138 <debug>\typeout{:: _check_lang:Nn~#1~/~#2~/~#3~/}
139   \bool_if:NTF \l_@@_never_check_bool
140     { \prg_return_true: }
141     {
142     \bool_if:NTF { \tl_if_empty_p:e {#3} }
143     { \prg_return_false: }
144     {
145 <*XE>
146       \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
147       \@@_iv_str_to_num:Nx \l_@@_script_int {#3}
148       \int_set:Nn \l_@@_tmpb_int
149         { \XeTeXOTcountlanguages #1 \l_@@_script_int }
150       \int_zero:N \l_@@_tmpa_int
151       \bool_set_false:N \l__fontspec_check_bool
152       \bool_until_do:nn { \int_compare_p:nNn \l_@@_tmpa_int = \l_@@_tmpb_int }

```

```

153     {
154       \int_set:Nn \l_@@_tmpc_int
155         { \XeTeXOTlanguagetag #1 \l_@@_script_int \l_@@_tmpa_int }
156
157       \int_compare:nNnTF \l_@@_tmpc_int = \l_@@_strnum_int
158         {
159           \bool_set_true:N \l__fontspec_check_bool
160           \int_set:Nn \l_@@_tmpa_int {\l_@@_tmpb_int}
161         }
162         {
163           \int_incr:N \l_@@_tmpa_int
164         }
165     }
166     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
167 </XE>
168 (*LU)
169     \@@_ot_validate_tag:x {#2}
170     \@@_ot_validate_tag:x {#3}
171     \cs_if_eq:NNTF #1 \font
172       { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
173       { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
174     \@@_lua_function:neee {check_ot_lang} {\l_@@_tmp_tl} {#2} {#3}
175     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
176 </LU>
177   }
178 }
179 }

```

(End of definition for \@@_check_lang:NnnTF and \@@_check_lang:NnTF. These functions are documented on page ??.)

\@@_check_ot_feat:NnTF This macro takes an OpenType feature tag and checks if it exists in the current font/script/language.
 \@@_check_ot_feat:NnnnTF \l_@@_strnum_int is used to store the number corresponding to the feature tag string. The script used is whatever's held in \l_@@_script_int. By default, that's the number corresponding to 'latn'. The language used is \l_@@_language_int, by default 0, the 'default language'.

```

180 \prg_new_conditional:Nnn \@@_check_ot_feat:Nn {TF,F}
181   {
182     \@@_check_ot_feat:NnnnTF #1 {#2} {\l_@@_lang_tl} {\l_@@_script_tl}
183     {\prg_return_true:} {\prg_return_false:}
184   }
185 \prg_new_conditional:Nnn \@@_check_ot_feat:Nnnn {TF,F}
186   {
187     \bool_if:NTF \l_@@_never_check_bool
188       { \prg_return_true: }
189     {
190       \bool_if:nTF { \tl_if_empty_p:e {#3} || \tl_if_empty_p:e {#4} }
191         { \prg_return_false: }
192         {
193 (*XE)
194 (debug)\typeout{::~ fontspec_check_ot_feat:nnn~ {#2}{#3}{#4}}
195       \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}

```

```

196
197 \str_if_eq:eeTF {#3} {dflt}
198 { \int_zero:N \l_@@_language_int }
199 { \@@_iv_str_to_num:Nx \l_@@_language_int {#3} }
200 \@@_iv_str_to_num:Nx \l_@@_script_int {#4}
201
202 \int_set:Nn \l_tmpb_int
203 { \XeTeXOTcountfeatures #1 \l_@@_script_int \l_@@_language_int }
204
205 \int_zero:N \l_tmpa_int
206 \bool_set_false:N \l_@@_check_bool
207 \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
208 {
209   \ifnum\XeTeXOTfeaturetag #1 \l_@@_script_int \l_@@_language_int
210     \l_tmpa_int =\l_@@_strnum_int
211     \bool_set_true:N \l_@@_check_bool
212     \int_set:Nn \l_tmpa_int {\l_tmpb_int}
213   \else
214     \int_incr:N \l_tmpa_int
215   \fi
216 }
217 \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
218 </XE>
219 <*LU>
220 <debug>\typeout{::~~ fontspec_check_ot_feat:n~ {#1}}
221 \@@_ot_validate_tag:x {#2}
222 \@@_ot_validate_tag:x {#3}
223 \@@_ot_validate_tag:x {#4}
224 \cs_if_eq:NNTF #1 \font
225 { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
226 { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
227 \@@_lua_function:neeee {check_ot_feat} {\l_@@_tmp_tl} {#2} {#3} {#4}
228 \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
229 </LU>
230 }
231 }
232 }

```

(End of definition for \@@_check_ot_feat:NnTF and \@@_check_ot_feat:NnnnTF. These functions are documented on page ??.)

1.2 OpenType feature information

```

233 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {aalt}{Access~All~Alternates}
234 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvf}{Above~base~Forms}
235 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvm}{Above~base~Mark~Positioning}
236 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvs}{Above~base~Substitutions}
237 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {afrc}{Alternative~Fractions}
238 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {akhn}{Akhands}
239 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwf}{Below~base~Forms}
240 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwm}{Below~base~Mark~Positioning}
241 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blws}{Below~base~Substitutions}

```

242 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {calt}{Contextual~Alternates}
243 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {case}{Case-Sensitive~Forms}
244 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ccmp}{Glyph~Composition~/~Decomposition}
245 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cfar}{Conjunct~Form~After~Ro}
246 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cjct}{Conjunct~Forms}
247 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {clig}{Contextual~Ligatures}
248 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cpct}{Centered~CJK~Punctuation}
249 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {csp}{Capital~Spacing}
250 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {csw}{Contextual~Swash}
251 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {curs}{Cursive~Positioning}
252 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cvNN}{Character~Variant~\$N\$}
253 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2pc}{Petite~Capitals~From~Capitals}
254 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2sc}{Small~Capitals~From~Capitals}
255 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dist}{Distances}
256 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dlig}{Discretionary~Ligatures}
257 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dnom}{Denominators}
258 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dtls}{Dotless~Forms}
259 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {expt}{Expert~Forms}
260 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {falt}{Final~Glyph~on~Line~Alternates}
261 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin2}{Terminal~Forms~\#2}
262 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin3}{Terminal~Forms~\#3}
263 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fina}{Terminal~Forms}
264 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {flac}{Flattened~accent~forms}
265 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {frac}{Fractions}
266 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fwid}{Full~Widths}
267 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {half}{Half~Forms}
268 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {haln}{Halant~Forms}
269 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {halt}{Alternate~Half~Widths}
270 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hist}{Historical~Forms}
271 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hkna}{Horizontal~Kana~Alternates}
272 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hlig}{Historical~Ligatures}
273 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hngl}{Hangul}
274 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hojo}{Hojo~Kanji~Forms}
275 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hwid}{Half~Widths}
276 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {init}{Initial~Forms}
277 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {isol}{Isolated~Forms}
278 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ital}{Italics}
279 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jalt}{Justification~Alternates}
280 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp78}{JIS78~Forms}
281 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp83}{JIS83~Forms}
282 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp90}{JIS90~Forms}
283 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp04}{JIS2004~Forms}
284 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {kern}{Kerning}
285 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lfb}{Left~Bounds}
286 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {liga}{Standard~Ligatures}
287 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ljmo}{Leading~Jamo~Forms}
288 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lnum}{Lining~Figures}
289 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {locl}{Localized~Forms}
290 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltra}{Left-to-right~alternates}
291 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltrm}{Left-to-right~mirrored~forms}
292 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mark}{Mark~Positioning}

293 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {med2}{Medial~Forms~\#2}
294 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {medi}{Medial~Forms}
295 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mgrk}{Mathematical~Greek}
296 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mkmk}{Mark~to~Mark~Positioning}
297 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mset}{Mark~Positioning~via~Substitution}
298 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nalt}{Alternate~Annotation~Forms}
299 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nlck}{NLC~Kanji~Forms}
300 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nukt}{Nukta~Forms}
301 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {numr}{Numerators}
302 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {onum}{Oldstyle~Figures}
303 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {opbd}{Optical~Bounds}
304 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ordn}{Ordinals}
305 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ornm}{Ornaments}
306 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {palt}{Proportional~Alternate~Widths}
307 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pcap}{Petite~Capitals}
308 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pkna}{Proportional~Kana}
309 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pnum}{Proportional~Figures}
310 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pref}{Pre~Base~Forms}
311 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pres}{Pre~base~Substitutions}
312 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstf}{Post~base~Forms}
313 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {psts}{Post~base~Substitutions}
314 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pwid}{Proportional~Widths}
315 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {qwid}{Quarter~Widths}
316 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rand}{Randomize}
317 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rclt}{Required~Contextual~Alternates}
318 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rkrf}{Rakar~Forms}
319 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rlig}{Required~Ligatures}
320 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rphf}{Reph~Forms}
321 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtbd}{Right~Bounds}
322 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtla}{Right~to~left~alternates}
323 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtlm}{Right~to~left~mirrored~forms}
324 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ruby}{Ruby~Notation~Forms}
325 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rvrn}{Required~Variation~Alternates}
326 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {salt}{Stylistic~Alternates}
327 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sinf}{Scientific~Inferiors}
328 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {size}{Optical~size}
329 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smcp}{Small~Capitals}
330 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smp1}{Simplified~Forms}
331 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssNN}{Stylistic~Set~\$N\$}
332 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssty}{Math~script~style~alternates}
333 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {stch}{Stretching~Glyph~Decomposition}
334 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {subs}{Subscript}
335 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sups}{Superscript}
336 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {swsh}{Swash}
337 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {titl}{Titling}
338 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing~Jamo~Forms}
339 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional~Name~Forms}
340 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular~Figures}
341 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {trad}{Traditional~Forms}
342 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third~Widths}
343 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {unic}{Unicase}

```

344 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate-Vertical-Metrics}
345 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu-Variants}
346 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical-Writing}
347 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate-Vertical-Half-Metrics}
348 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vjmo}{Vowel-Jamo-Forms}
349 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical-Kana-Alternates}
350 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkrn}{Vertical-Kerning}
351 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional-Alternate-Vertical-Me}
352 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrt2}{Vertical-Alternates-and-Rotation}
353 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrtr}{Vertical-Alternates-for-Rotation}
354 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {zero}{Slashed-Zero}

```

TODO: move the above elsewhere!!

File XII

fontspec-code-graphite.dtx

1 Graphite/AAT code

`\@@_define_aat_feature_group:n`

```
1 \cs_new:Nn \@@_define_aat_feature_group:n
2 {
3   \keys_define:nn {fontspec-aat} { #1 .multichoice: }
4 }
```

(End of definition for \@@_define_aat_feature_group:n. This function is documented on page ??.)

`\@@_define_aat_feature:nmnn`

```
5 \cs_new:Nn \@@_define_aat_feature:nmnn
6 {
7   \keys_define:nn {fontspec-aat}
8     {
9       #1/#2 .code:n = { \@@_make_AAT_feature:nn {#3}{#4} }
10    }
11 }
```

(End of definition for \@@_define_aat_feature:nmnn. This function is documented on page ??.)

`\@@_make_AAT_feature:nn`

```
12 \cs_new:Nn \@@_make_AAT_feature:nn
13 {
14   \tl_if_empty:nTF {#1}
15     { \@@_warning:n {aat-feature-not-exist} }
16     {
17       \exp_args:No \@@_make_AAT_feature_string:NmnTF \l_@@_fontface_cs_tl {#1} {#2}
18       {
19         \@@_update_featstr:n {\l_fontspec_feature_string_tl}
20       }
21       {
22         \@@_warning:nx {aat-feature-not-exist-in-font} {#1,#2}
23       }
24     }
25 }
```

(End of definition for \@@_make_AAT_feature:nn. This function is documented on page ??.)

`\@@_make_AAT_feature_string:NmnTF`

This macro takes the numerical codes for a font feature and creates a specified macro containing the string required in the font definition to turn that feature on or off. Used primarily in [...], but also used to check if small caps exists in the requested font (see page 59).

For exclusive selectors, it's easy; just grab the string: For *non*-exclusive selectors, it's a little more complex. If the selector is even, it corresponds to switching the feature on. If the selector is *odd*, it corresponds to switching the feature off. But X_YL_AT_EX doesn't return a selector string for this number, since the feature is defined for the 'switching on' value. So we need to

check the selector of the previous number, and then prefix the feature string with ! to denote the switch.

Finally, save out the complete feature string in `\l_fontspec_feature_string_tl`.

```

26 \prg_new_conditional:Nnn \@@_make_AAT_feature_string:Nnn {TF,T,F}
27 {
28   \tl_set:Nx \l_@@_tmpa_tl { \XeTeXfeaturename #1 #2 }
29   \tl_if_empty:NTF \l_@@_tmpa_tl
30     { \prg_return_false: }
31     {
32       \int_compare:nTF { \XeTeXisexclusivefeature #1 #2 > 0 }
33         {
34           \tl_set:Nx \l_@@_tmpb_tl { \XeTeXselectorname #1 #2\space #3}
35         }
36         {
37           \int_if_even:nTF {#3}
38             {
39               \tl_set:Nx \l_@@_tmpb_tl { \XeTeXselectorname #1 #2\space #3}
40             }
41             {
42               \tl_set:Nx \l_@@_tmpb_tl
43                 {
44                   \XeTeXselectorname #1 #2\space \numexpr#3-1\relax
45                 }
46               \tl_if_empty:NF \l_@@_tmpb_tl { \tl_put_left:Nn \l_@@_tmpb_tl {!} }
47             }
48         }
49
50       \tl_if_empty:NTF \l_@@_tmpb_tl
51         { \prg_return_false: }
52         {
53           \tl_set:Nx \l_fontspec_feature_string_tl { \l_@@_tmpa_tl = \l_@@_tmpb_tl }
54           \prg_return_true:
55         }
56     }
57 }

```

(End of definition for `\@@_make_AAT_feature_string:NnnTF`. This function is documented on page ??.)

File XIII

fontspec-code-keyval.dtx

1 Font loading (keyval) definitions

This package uses a large number of keyval modules which operate sequentially on keyval input to ensure priority.

```
1 \clist_gset:Nn \g_@@_all_keyval_modules_clist
2 {
3   fontspec, fontspec-opentype, fontspec-aat,
4   fontspec-preparse, fontspec-preparse-cfg, fontspec-preparse-external, fontspec-preparse-ne
5   fontspec-renderer
6 }
```

Wrapper function to save some characters in the source:

```
7 \cs_new:Nn \@@_keys_define_code:nnn
8 {
9   \keys_define:nn {#1} { #2 .code:n = {#3} }
10 }
```

For catching features that cannot be used in `\addfontfeatures`:

```
11 \cs_new:Nn \@@_aff_error:n
12 {
13   \@@_keys_define_code:nnn {fontspec-addfontfeatures} {#1}
14   { \@@_error:nx {not-in-addfontfeatures} {#1} }
15 }
```

1.1 Pre-pre-parsing stages

These features are extracted from the font feature list before all others.

Don't load font config file

```
16 \@@_keys_define_code:nnn {fontspec-preparse-cfg} {IgnoreFontspecFile}
17 {
18   \bool_set_false:N \l_@@_fontcfg_bool
19 }
20 \@@_keys_define_code:nnn {fontspec-preparse-external} {IgnoreFontspecFile}
21 {
22   \bool_set_false:N \l_@@_fontcfg_bool
23 }
```

Path For fonts that aren't installed in the system. If no argument is given, the font is located with `kpsewhich`; it's either in the current directory or the \TeX tree. Otherwise, the argument given defines the file path of the font.

```
24 \@@_keys_define_code:nnn {fontspec-preparse-external} {Path}
25 {
26   \bool_set_true:N \l_@@_nobf_bool
27   \bool_set_true:N \l_@@_noit_bool
28   \bool_set_true:N \l_@@_external_bool
```

```

29     \tl_set:Nn \l_@@_font_path_tl {#1}
30     \bool_lazy_and:nnTF { \l_@@_external_kpse_bool } { \tl_if_empty_p:N \l_@@_font_path_tl }
31     {
32         \@@_font_is_kpse:
33     }
34     {
35         \@@_font_is_file:
36     }
37 <*XE>
38     \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
39 </XE>
40 }
41 \aliasfontfeature{Path}{ExternalLocation}
42 \@@_keys_define_code:nnn {fontspec} {Path} {}

```

(End of definition for Path. This function is documented on page ??.)

Extension For fonts that aren't installed in the system. Specifies the font extension to use.

```

43 \@@_keys_define_code:nnn {fontspec-prepare-external} {Extension}
44 {
45     \tl_set:Nn \l_@@_extension_tl {#1}
46     \bool_if:NF \l_@@_external_bool
47     {
48         \keys_set:nn {fontspec-prepare-external} {Path}
49     }
50 }
51 \tl_clear:N \l_@@_extension_tl
52 \@@_keys_define_code:nnn {fontspec} {Extension} {}

```

KpseOnly If the font is specified by filename, only search for it through kpse. Xe_{La}TeX does not support finding system fonts by filename so this is always implicitly set there.

```

53 \@@_keys_define_code:nnn {fontspec-prepare-external} {KpseOnly}
54 {
55     \bool_set_true:N \l_@@_external_kpse_bool
56     \bool_if:NT \l_@@_external_bool
57     {
58         \@@_font_is_kpse:
59     }
60 }
61 \@@_keys_define_code:nnn {fontspec} {KpseOnly} {}

```

Renderer This feature must be processed before all others (the other font shape and features options are also pre-parsed for convenience) because the renderer determines the format of the features and whether certain features are available.

```

62 <*XE>
63 \keys_define:nn {fontspec-renderer}
64 {
65     Renderer .choices:nn =
66         {AAT,ICU,OpenType,Graphite,Full,Basic,Node,Base,HarfBuzz,Harfbuzz}
67     {

```

```

68     \int_compare:nTF {\l_keys_choice_int <= 4}
69     {
70         \tl_set:Nx \l_@@_renderer_tl
71         {
72             \int_case:nn \l_keys_choice_int { 1 {/AAT} 2 {/OT} 3 {/OT} 4 {/GR} }
73         }
74 <debug>\typeout{Renderer:~ \l_@@_renderer_tl}
75         \tl_gset:Nx \g_@@_single_feat_tl { \l_@@_renderer_tl }
76     }
77     {
78         \@@_warning:nx {only-luatex-feature} {Renderer=Full/Basic/Node/Base/HarfBuzz}
79     }
80 }
81 }
82 </XE>
83 *LU
84 \keys_define:nn {fontspec-renderer}
85 {
86     Renderer .choices:nn =
87     {Full,Node,Basic,Base,HarfBuzz,Harfbuzz,OpenType,AAT,Graphite}
88     {
89         \int_compare:nT {\l_keys_choice_int >= 5} { \bool_set_true:N \l_@@_harfbuzz_bool }
90
91         \tl_set:Nx \l_@@_mode_tl
92         {
93             \int_case:nn \l_keys_choice_int { 1 {node} 2 {node} 3 {base} 4 {base} 5 {harf} 6 {
94         }
95
96         \tl_set:Nx \l_@@_shaper_tl
97         {
98             \int_case:nn \l_keys_choice_int { 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {ot} 8 {coretext
99         }
100
101 <debug>\typeout{Mode:~"\l_@@_mode_tl"~/~Shaper:~"\l_@@_shaper_tl"}
102
103         \tl_gset:Nx \g_@@_single_feat_tl
104         {
105             mode=\l_@@_mode_tl ;
106             \tl_if_empty:NF \l_@@_shaper_tl { shaper=\l_@@_shaper_tl}
107         }
108     } ,
109
110     Renderer unknown .code:n =
111     {
112         \bool_set_true:N \l_@@_harfbuzz_bool
113         \@@_warning:nx {unknown-renderer} {#1}
114         \tl_set:Nn \l_@@_mode_tl {harf}
115         \tl_set:Nn \l_@@_shaper_tl {#1}
116     } ,
117 }
118 </LU>

```

1.2 Pre-parsed features

OpenType script/language See later for the resolutions from fontspec features to OpenType definitions.

```
119 \@@_keys_define_code:nmn {fontspec-preparse} {Script}
120   {
121   (XE)   \keys_set:nm {fontspec-renderer} {Renderer=OpenType}
122         \tl_set:Nn \l_@@_script_name_tl {#1}
123   }
```

Exactly the same:

```
124 \@@_keys_define_code:nmn {fontspec-preparse} {Language}
125   {
126   (XE)   \keys_set:nm {fontspec-renderer} {Renderer=OpenType}
127         \tl_set:Nn \l_@@_lang_name_tl {#1}
128   }
```

TTC font index

```
129 \@@_keys_define_code:nmn {fontspec-preparse} {FontIndex}
130   {
131     \str_if_eq:eeF { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
132     { \@@_warning:n {font-index-needs-ttc} }
133   (XE)   \tl_set:Nn \l_@@_ttc_index_tl {:#1}
134   (LU)   \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
135   }
136 \@@_keys_define_code:nmn {fontspec} {FontIndex}
137   {
138   (XE)   \tl_set:Nn \l_@@_ttc_index_tl {:#1}
139   (LU)   \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
140   }
```

1.3 Font faces

Upright

```
141 \@@_keys_define_code:nmn {fontspec-preparse-external} {UprightFont}
142   {
143     \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
144   }
```

Italic and slanted

```
145 \@@_keys_define_code:nmn {fontspec-preparse-external} {ItalicFont}
146   {
147     \tl_if_empty:nTF {#1}
148     {
149       \bool_set_true:N \l_@@_noit_bool
150     }
151     {
152       \bool_set_false:N \l_@@_noit_bool
153       \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
154     }
155   }
```

```

156 \@@_keys_define_code:nmn {fontspec-prepare-external} {SlantedFont}
157   {
158     \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
159   }
160 \@@_keys_define_code:nmn {fontspec-prepare-external} {SwashFont}
161   {
162     \fontspec_complete_fontname:Nn \l_@@_fontname_sw_tl {#1}
163   }

```

Bold (NFSS) Series By default, fontspec uses the default bold series, `\bfdefault`. We want to be able to make this extensible. This code is not yet functional!

```

164 %\@@_keys_define_code:nmn {fontspec-prepare-external} {BoldSeries}
165 % {
166 %   \tl_gset:Nx \g_@@_curr_series_tl { #1 }
167 %   \seq_put_right:Nx \l_@@_bf_series_seq { #1 }
168 % }

```

Bold This contains some stubb code to allow more than one bold font to be loaded.

```

169 \@@_keys_define_code:nmn {fontspec-prepare-external} {BoldFont}
170   {
171     \tl_if_empty:nTF {#1}
172       {
173         \bool_set_true:N \l_@@_nobf_bool
174       }
175       {
176         \bool_set_false:N \l_@@_nobf_bool
177         \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
178
179         \seq_if_empty:NT \l_@@_bf_series_seq
180           {
181             \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
182             \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
183           }
184
185         \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
186           {
187             \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
188           }
189
190         \prop_put:NxV \l_@@_nfss_prop {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
191
192         (debug)\typeout{Setting-bold-font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
193
194       }
195   }

```

Bold italic/slanted

```

196 \@@_keys_define_code:nmn {fontspec-prepare-external} {BoldItalicFont}
197   {

```

```

198     \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
199   }
200 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSlantedFont}
201   {
202     \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
203   }
204 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSwashFont}
205   {
206     \fontspec_complete_fontname:Nn \l_@@_fontname_bfsw_tl {#1}
207   }

```

Small caps Small caps isn't pre-parsed because it can vary with others above:

```

208 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
209   {
210     \tl_if_empty:nTF {#1}
211       {
212         \bool_set_true:N \l_@@_nosc_bool
213       }
214       {
215         \bool_set_false:N \l_@@_nosc_bool
216         \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
217       }
218   }

```

1.3.1 Prepared font features

```

219 \@@_keys_define_code:nnn {fontspec-preparse} {UprightFeatures}
220   {
221     \clist_put_right:Nn \l_@@_fontfeat_up_clist {#1}
222   }
223 \@@_keys_define_code:nnn {fontspec-preparse} {BoldFeatures}
224   {
225     \clist_put_right:Nn \l_@@_fontfeat_bf_clist {#1}
226     % \prop_put:NxV \l_@@_nfss_prop
227     %   {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
228   }
229 \@@_keys_define_code:nnn {fontspec-preparse} {ItalicFeatures}
230   {
231     \clist_put_right:Nn \l_@@_fontfeat_it_clist {#1}
232   }
233 \@@_keys_define_code:nnn {fontspec-preparse} {BoldItalicFeatures}
234   {
235     \clist_put_right:Nn \l_@@_fontfeat_bfit_clist {#1}
236   }
237 \@@_keys_define_code:nnn {fontspec-preparse} {SlantedFeatures}
238   {
239     \clist_put_right:Nn \l_@@_fontfeat_sl_clist {#1}
240   }
241 \@@_keys_define_code:nnn {fontspec-preparse} {BoldSlantedFeatures}
242   {

```

```

243     \clist_put_right:Nn \l_@@_fontfeat_bfsl_clist {#1}
244   }
245   \@@_keys_define_code:nmn {fontspec-preparse} {SwashFeatures}
246   {
247     \clist_put_right:Nn \l_@@_fontfeat_sw_clist {#1}
248   }
249   \@@_keys_define_code:nmn {fontspec-preparse} {BoldSwashFeatures}
250   {
251     \clist_put_right:Nn \l_@@_fontfeat_bfsw_clist {#1}
252   }

```

Note that small caps features can vary by shape, so these in fact *aren't* pre-parsed.

```

253   \@@_keys_define_code:nmn {fontspec} {SmallCapsFeatures}
254   {
255     \bool_if:NF \l_@@_firsttime_bool
256     {
257       \clist_put_right:Nn \l_@@_fontfeat_sc_clist {#1}
258     }
259   }

```

Features varying by size

```

260   \@@_keys_define_code:nmn {fontspec-preparse} {SizeFeatures}
261   {
262     \clist_set:Nn \l_@@_sizefeat_clist {#1}
263     \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
264   }
265   \@@_keys_define_code:nmn {fontspec-preparse-nested} {SizeFeatures}
266   {
267     \clist_set:Nn \l_@@_sizefeat_clist {#1}
268     \tl_if_empty:NT \l_@@_this_font_tl
269     { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
270   }
271   \@@_keys_define_code:nmn {fontspec-preparse-nested} {Font}
272   {
273     \tl_set:Nn \l_@@_this_font_tl {#1}
274   }
275   \@@_keys_define_code:nmn {fontspec} {SizeFeatures}
276   {
277     % dummy
278   }
279   \@@_keys_define_code:nmn {fontspec} {Font}
280   {
281     % dummy
282   }
283   \@@_keys_define_code:nmn {fontspec-sizing} {Size}
284   {
285     \tl_set:Nn \l_@@_size_tl {#1}
286   }
287   \@@_keys_define_code:nmn {fontspec-sizing} {Font}
288   {
289     \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
290   }

```


A hack to fix a test, needs to be investigated why necessary!

```
297 \@@_keys_define_code:nmn {fontspec-opentype} {UprightFont} {}
298 \@@_keys_define_code:nmn {fontspec-opentype} {ItalicFont} {}
299 \@@_keys_define_code:nmn {fontspec-opentype} {SlantedFont} {}
300 \@@_keys_define_code:nmn {fontspec-opentype} {BoldFont} {}
301 \@@_keys_define_code:nmn {fontspec-opentype} {BoldItalicFont} {}
302 \@@_keys_define_code:nmn {fontspec-opentype} {BoldSlantedFont} {}
```

1.4 General font-independent features

These features can be applied to any font.

NFSS encoding For the very brave.

```
297 \@@_keys_define_code:nmn {fontspec-prepare} {NFSSEncoding}
298 {
299   \tl_gset:Nx \g_@@_nfss_enc_tl { #1 }
300 }
```

NFSS family Interactions with other packages will sometimes require setting the NFSS family explicitly. (By default fontspec auto-generates one based on the font name.)

```
301 \@@_keys_define_code:nmn {fontspec-prepare} {NFSSFamilY}
302 {
303   \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
304 }
```

NFSS series/shape This option looks similar in name but has a very different function.

```
305 \@@_keys_define_code:nmn {fontspec-prepare} {FontFace}
306 {
307   \tl_clear:N \l_@@_this_font_tl
308   \clist_set:No \l_@@_arg_clist { \use_iii:nmn #1 }
309   \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
310   \int_compare:nT { \clist_count:N \l_@@_arg_clist = 1 }
311   {
312     <debug>\typeout{FontFace~ parsing:~ one~ clist~ item}
313     \tl_if_in:NnF \l_@@_arg_clist {=}
314     {
315       <debug>\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
316       \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
317       \tl_clear:N \l_@@_this_feat_clist
318     }
319   }
320
321   \@@_add_nfssfont:nnnn
322   {\use_i:nmn #1} {\use_ii:nmn #1} {\l_@@_this_font_tl} {\l_@@_this_feat_clist}
323 }
```

Scale If the input isn't one of the pre-defined string options, then it's gotta be numerical. `\fontspec_calc_scale:n` and `\fontspec_calc_scale:nn` do all the work in the auto-scaling cases.

```

324 \@@_keys_define_code:nnn {fontspec} {Scale}
325   {
326     \str_case:nnF {#1}
327     {
328       {MatchLowercase} { \@@_calc_scale:n {5} }
329       {MatchUppercase} { \@@_calc_scale:n {8} }
330       {MatchAveragecase} { \@@_calc_scale:nn {5} {8} }
331     }
332     { \tl_set:Nx \l_@@_scale_tl {#1} }
333     \@@_info:n {set-scale}
334   }

```

ScaleAgain

```

335 \@@_keys_define_code:nnn {fontspec} {ScaleAgain}
336   {
337     \tl_if_empty:NT \l_@@_scale_tl { \tl_set:Nn \l_@@_scale_tl {1} }
338     \tl_set:Nx \l_@@_scale_tl { \fp_eval:n { #1 * \l_@@_scale_tl } }
339     \@@_info:n {set-scale}
340   }

```

`\@@_calc_scale:n` This macro calculates the amount of scaling between the default roman font and the (default shape of) the font being selected such that the font dimension that is input is equal for both. The only font dimensions that justify this are 5 (lowercase height) and 8 (uppercase height in X_YT_EX).

This script is executed for every extra shape, which seems wasteful, but allows alternate italic shapes from a separate font, say, to be loaded and to be auto-scaled correctly. Even if this would be ugly.

To begin, change to `\rmfamily` but use internal commands in case `csermfamily` has been overwritten. (Note that changing `\rmfamily` with `fontspec` resets `\encodingdefault` appropriately.)

```

341 \cs_new:Nn \@@_calc_scale:n
342   {
343     \group_begin:
344
345     \fontencoding { \encodingdefault }
346     \fontfamily { \familydefault }
347     \selectfont
348
349     \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
350     \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_@@_fontface_cs_tl
351
352     \tl_set:Nx \l_@@_scale_tl
353     {
354       \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
355                   \dim_to_fp:n {\l_@@_tmpb_dim} }
356     }
357

```

```

358     \exp_args:NNNx
359     \group_end:
360     \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
361 }

```

(End of definition for \@@_calc_scale:n. This function is documented on page ??.)

\@@_calc_scale:nn This macro calls `\fontspec_calc_scale:n` twice and then sets the scale to the average of the two results.

```

362 \cs_new:Nn \@@_calc_scale:nn
363 {
364     \group_begin:
365     \__fontspec_calc_scale:n {#1}
366     \tl_set_eq:NN \l_@@_tmp_tl \l_@@_scale_tl
367     \__fontspec_calc_scale:n {#2}
368     \tl_set:Nx \l_@@_scale_tl
369     {
370         \fp_eval:n { (\l_@@_tmp_tl + \l_@@_scale_tl)/2 }
371     }
372     \exp_args:NNNx
373     \group_end:
374     \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
375 }

```

(End of definition for \@@_calc_scale:nn. This function is documented on page ??.)

\@@_set_font_dimen:NnN This function sets the dimension #1 (for font #3) to 'fontdimen' #2 for either font dimension 5 (x-height) or 8 (cap-height). If, for some reason, these return an incorrect 'zero' value (as `\fontdimen8` might for a .tfm font), then we cheat and measure the height of a glyph. We assume in this case that the font contains either an 'X' or an 'x'.

```

376 \cs_new:Nn \@@_set_font_dimen:NnN
377 {
378     \dim_set:Nn #1 { \fontdimen #2 #3 }
379     \dim_compare:nNnT #1 = {0pt}
380     {
381         \settoheight #1
382         {
383             \str_if_eq:nnTF {#3} {\font} \rmfamily #3
384             \int_case:nnF #2
385             {
386                 {5} {x} % x-height
387                 {8} {X} % cap-height
388             } {?} % "else" clause; never reached.
389         }
390     }
391 }

```

(End of definition for \@@_set_font_dimen:NnN. This function is documented on page ??.)

Inter-word space These options set the relevant `\fontdimens` for the font being loaded.

```

392 \@@_keys_define_code:nmn {fontspec} {WordSpace}
393   {
394     \bool_if:NF \l_@@_firsttime_bool
395       { \_fontspec_parse_wordspace:w #1,,,\q_stop }
396   }
397 \@@_aff_error:n {WordSpace}

```

`_fontspec_parse_wordspace:w` This macro determines if the input to `WordSpace` is of the form `{X}` or `{X,Y,Z}` and executes the font scaling. If the former input, it executes `{X,X,X}`.

```

398 \cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
399   {
400     \tl_if_empty:nTF {#4}
401       {
402         \tl_set:Nn \l_@@_wordspace_adjust_tl
403           {
404             \fontdimen 2 \font = #1 \fontdimen 2 \font
405             \fontdimen 3 \font = #1 \fontdimen 3 \font
406             \fontdimen 4 \font = #1 \fontdimen 4 \font
407           }
408       }
409     {
410       \tl_set:Nn \l_@@_wordspace_adjust_tl
411         {
412           \fontdimen 2 \font = #1 \fontdimen 2 \font
413           \fontdimen 3 \font = #2 \fontdimen 3 \font
414           \fontdimen 4 \font = #3 \fontdimen 4 \font
415         }
416     }
417 }

```

(End of definition for `_fontspec_parse_wordspace:w`. This function is documented on page ??.)

Punctuation space Scaling factor for the nominal `\fontdimen#7`.

```

418 \@@_keys_define_code:nmn {fontspec} {PunctuationSpace}
419   {
420     \str_case_e:nnF {#1}
421       {
422         {WordSpace}
423         {
424           \tl_set:Nn \l_@@_punctspace_adjust_tl
425             { \fontdimen 7 \font = 0 \fontdimen 2 \font }
426         }
427         {TwiceWordSpace}
428         {
429           \tl_set:Nn \l_@@_punctspace_adjust_tl
430             { \fontdimen 7 \font = 1 \fontdimen 2 \font }
431         }
432       }
433     {
434       \tl_set:Nn \l_@@_punctspace_adjust_tl

```

```

435         { \fontdimen 7 \font = #1 \fontdimen 7 \font }
436     }
437 }
438 \@@_aff_error:n {PunctuationSpace}

```

Secret hook into the font-adjustment code

```

439 \@@_keys_define_code:nmn {fontspec} {FontAdjustment}
440 {
441     \tl_put_right:Nx \l_@@_postadjust_tl {#1}
442 }

```

Letterspacing

```

443 \@@_keys_define_code:nmn {fontspec} {LetterSpace}
444 {
445     \@@_update_featstr:n {letterspace=#1}
446 }

```

Hyphenation character This feature takes one of three arguments: ‘None’, *⟨glyph⟩*, or *⟨slot⟩*. If the input isn’t the first, and it’s one character, then it’s the second; otherwise, it’s the third.

LuaTeX decouples hyphenation from font settings, so only `HyphenChar=None` works for that engine.

```

447 \@@_keys_define_code:nmn {fontspec} {HyphenChar}
448 {
449     \str_if_eq:nnTF {#1} {None}
450     {
451         \tl_put_right:Nn \l_@@_postadjust_tl
452         { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
453     }
454     {
455     (LU)      \@@_warning:nx {only-xetex-feature} {HyphenChar}
456
457         \tl_if_single:nTF {#1}
458         { \tl_set:Nn \l_@@_hyphenchar_tl {`#1} }
459         { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }
460
461         \exp_args:No \@@_primitive_font_glyph_if_exist:NnTF \l_@@_fontface_cs_tl {\l_@@_hyphen
462         {
463             \tl_put_right:Nn \l_@@_postadjust_tl
464             { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
465         }
466         { \@@_error:nxx {no-glyph}{\l_fontspec_fontname_tl}{#1} }
467     }
468 }
469 }
470 \@@_aff_error:n {HyphenChar}

```

Color Test first if the color is a named `l3color`, then if it is a color from `xcolor`, which names its colours `\color@<name>`. If this fails the argument is assumed to be a hex color.

```

471 \@@_keys_define_code:nmn {fontspec} {Color}

```

```

472 {
473 (*XE)
474 \color_if_exist:nTF {#1}
475 {
476 \color_export:nnN {#1} {HTML}\l_@@_hexcol_tl
477 }
478 {
479 \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
480 {
481 \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
482 }
483 {
484 \int_compare:nTF { \tl_count:n {#1} == 6 }
485 { \tl_set:Nn \l_@@_hexcol_tl {#1} }
486 {
487 \int_compare:nTF { \tl_count:n {#1} == 8 }
488 { \fontspec_parse_colour:viii #1 }
489 {
490 \bool_if:NF \l_@@_firsttime_bool
491 { \@@_warning:nx {bad-colour} {#1} }
492 }
493 }
494 }
495 }
496 (/XE)
497 (*LU)
498 \color_if_exist:nTF {#1}
499 {
500 \tl_set:Nn \l_@@_hexcol_tl {#1}
501 }
502 {
503 \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
504 {
505 \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
506 }
507 {
508 \int_compare:nTF { \tl_count:n {#1} == 6 }
509 { \tl_set:Nn \l_@@_hexcol_tl {#1} }
510 {
511 \int_compare:nTF { \tl_count:n {#1} == 8 }
512 { \fontspec_parse_colour:viii #1 }
513 {
514 \bool_if:NF \l_@@_firsttime_bool
515 { \@@_warning:nx {bad-colour} {#1} }
516 }
517 }
518 }
519 }
520 (/LU)
521 }
522 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8

```

```

523 {
524   \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
525   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
526     {
527       \bool_if:NF \l_@@_firsttime_bool
528         { \@@_warning:nx {opa-twice-col} {#7#8} }
529     }
530   \tl_set:Nn \l_@@_opacity_tl {#7#8}
531 }
532 \aliasfontfeature{Color}{Colour}
533 \@@_keys_define_code:nmn {fontspec} {Opacity}
534 {
535   \int_set:Nn \l_@@_tmp_int {255}
536   \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
537   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
538     {
539       \bool_if:NF \l_@@_firsttime_bool
540         { \@@_warning:nx {opa-twice} {#1} }
541     }
542   \tl_set:Nx \l_@@_opacity_tl
543     {
544     (LU)      ,
545               \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
546               \int_to_hex:n { \l_@@_tmp_int }
547             }
548   }

```

Mapping

```

549 (*XE)
550 \@@_keys_define_code:nmn {fontspec-aat} {Mapping}
551 {
552   \tl_set:Nn \l_@@_mapping_tl { #1 }
553 }
554 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
555 {
556   \tl_set:Nn \l_@@_mapping_tl { #1 }
557 }
558 (/XE)
559 (*LU)
560 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
561 {
562   \str_if_eq:nnTF {#1} {tex-text}
563     {
564       \@@_warning:n {no-mapping-ligtex}
565       \msg_redirect_name:nmn {fontspec} {no-mapping-ligtex} {none}
566       \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
567     }
568   { \@@_warning:n {no-mapping} }
569 }
570 (/LU)

```

1.4.1 Continuous font axes

```
571 <*XE>
572 \@@_keys_define_code:nmn {fontspec} {Weight}
573 {
574   \@@_update_featstr:n{weight=#1}
575 }
576 </XE>
577 <LU>\@@_define_opentype_variation_axis:nn {Weight} {wght}
578 <*XE>
579 \@@_keys_define_code:nmn {fontspec} {Width}
580 {
581   \@@_update_featstr:n{width=#1}
582 }
583 </XE>
584 <LU>\@@_define_opentype_variation_axis:nn {Width} {wdth}
585 \@@_define_opentype_variation_axis:nn {Slant} {slnt}
586 \@@_keys_define_code:nmn {fontspec} {OpticalSize}
587 <*XE>
588 {
589   \bool_if:NTF \l_@@_ot_bool
590   {
591     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
592   }
593   {
594     \bool_if:NT \l_@@_mm_bool
595     {
596       \@@_update_featstr:n { optical size = #1 }
597     }
598   }
599   \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
600   {
601     \bool_if:NT \l_@@_firsttime_bool
602     { \@@_warning:nx {no-opticals} {\l_fontspeg_fontname_tl} }
603   }
604 }
605 </XE>
606 <*LU>
607 {
608   \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
609 }
610 </LU>
```

For other potentially font specific variation axes, there is a raw setter available:

```
611 \@@_keys_define_code:nmn {fontspec-opentype} {RawAxis}
612 {
613   \prop_gput_from_keyval:Nn \g_@@_rawvariations_prop {#1}
614 }
```

1.4.2 Variation instances

```
615 \@@_keys_define_code:nmn {fontspec-opentype} {Instance}
616 {
617   \tl_gset:Nn \g_@@_instance_tl {#1}
```



```
618 }
```

1.4.3 Font transformations

These are to be specified to apply directly to a font shape:

```
619 \keys_define:nn {fontspec}
620 {
621   FakeSlant .code:n =
622   {
623     \@@_update_featstr:n {slant=#1}
624   },
625   FakeSlant .default:n = {0.2}
626 }
627 \keys_define:nn {fontspec}
628 {
629   FakeStretch .code:n =
630   {
631     \@@_update_featstr:n {extend=#1}
632   },
633   FakeStretch .default:n = {1.2}
634 }
635 \keys_define:nn {fontspec}
636 {
637   FakeBold .code:n =
638   {
639     \@@_update_featstr:n {embolden=#1}
640   },
641   FakeBold .default:n = {1.5}
642 }
```

These are to be given to a shape that has no real bold/italic to signal that fontspec should automatically create ‘fake’ shapes.

The behaviour is currently that only if both `AutoFakeSlant` and `AutoFakeBold` are specified, the bold italic is also faked.

These features presently *override* real shapes found in the font; in the future I’d like these features to be ignored in this case, instead. (This is just a bit harder to program in the current design of fontspec.)

```
643 \keys_define:nn {fontspec}
644 {
645   AutoFakeSlant .code:n =
646   {
647     \bool_if:NT \l_@@_firsttime_bool
648     {
649       \tl_set:Nn \l_@@_fake_slant_tl {#1}
650       \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
651       \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
652       \bool_set_false:N \l_@@_noit_bool
653
654       \tl_if_empty:NF \l_@@_fake_embolden_tl
655       {
656         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
657         {FakeBold=\l_@@_fake_embolden_tl}
```

```

658         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
659         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
660     }
661 }
662 },
663 AutoFakeSlant .default:n = {0.2}
664 }

```

Same but reversed:

```

665 \keys_define:nn {fontspec}
666 {
667     AutoFakeBold .code:n =
668     {
669         \bool_if:NT \l_@@_firsttime_bool
670         {
671             \tl_set:Nn \l_@@_fake_embolden_tl {#1}
672             \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
673             \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
674             \bool_set_false:N \l_@@_nobf_bool
675
676             \tl_if_empty:NF \l_@@_fake_slant_tl
677             {
678                 \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
679                 {FakeSlant=\l_@@_fake_slant_tl}
680                 \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
681                 \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
682             }
683         }
684     },
685     AutoFakeBold .default:n = {1.5}
686 }

```

1.4.4 Raw feature string

This allows savvy X_YTeX-ers to input font features manually if they have already memorised the OpenType abbreviations and don't mind not having error checking.

```

687 \@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}
688 {
689     \@@_update_featstr:n {#1}
690 }
691 \@@_keys_define_code:nnn {fontspec-aat} {RawFeature}
692 {
693     \@@_update_featstr:n {#1}
694 }

```

File XIV

fontspec-code-feat-opentype.dtx

1 OpenType feature definitions

```
1 \@@_feat_prop_add:mn {salt} { Alternate\,=\,$N$ }
2 \@@_feat_prop_add:mn {nalt} { Annotation\,=\,$N$ }
3 \@@_feat_prop_add:mn {ornm} { Ornament\,=\,$N$ }
4 \@@_feat_prop_add:mn {cvNN} { CharacterVariant\,=\,$N$:$M$ }
5 \@@_feat_prop_add:mn {ssNN} { StylisticSet\,=\,$N$ }
```

2 Regular key=val / tag definitions

2.1 Ligatures

```
6 \@@_define_opentype_feature_group:n {Ligatures}
7 \@@_define_opentype_feature:nnnnn {Ligatures} {ResetAll} {} {}
8 {
9   +dlig,-dlig,+rlig,-rlig,+liga,-liga,+dlig,-dlig,+clig,-clig,+hlig,-hlig,
10  <XE> mapping = tex-text
11  <LU> +tlig,-tlig
12 }
13 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Required} {rlig} {rlig} {}
14 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Common} {liga} {liga} {}
15 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Rare} {dlig} {dlig} {}
16 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Discretionary} {dlig} {dlig} {}
17 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Contextual} {clig} {clig} {}
18 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Historic} {hlig} {hlig} {}
```

Emulate CM extra ligatures.

```
19 <*XE>
20 \keys_define:nn {fontspec-opentype}
21 {
22   Ligatures / TeX .code:n = { \tl_set:Nn \l_@@_mapping_tl {tex-text} },
23   Ligatures / TeXOff .code:n = { \tl_clear:N \l_@@_mapping_tl },
24   Ligatures / TeXReset .code:n = { \tl_clear:N \l_@@_mapping_tl },
25 }
26 </XE>
27 <LU>\@@_define_opentype_onoffreset:nnnnn {Ligatures} {TeX} {} {tlig} {}
```

2.2 Letters

```
28 \@@_define_opentype_feature_group:n {Letters}
29 \@@_define_opentype_feature:nnnnn {Letters} {ResetAll} {} {}
30 {
31   +case,+smcp,+pcap,+c2sc,+c2pc,+unic,+rand,
32   -case,-smcp,-pcap,-c2sc,-c2pc,-unic,-rand
33 }
34 \@@_define_opentype_onoffreset:nnnnn {Letters} {Uppercase} {case} {case} {}
35 \@@_define_opentype_onoffreset:nnnnn {Letters} {SmallCaps} {smcp} {smcp} {+pcap,+unic}
```

```

36 \@@_define_opentype_onoffreset:nnnnn {Letters} {PetiteCaps} {pcap} {pcap} {+smcp,+unic}
37 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercaseSmallCaps} {c2sc} {c2sc} {+c2pc,+unic}
38 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercasePetiteCaps} {c2pc} {c2pc} {+c2sc,+unic}
39 \@@_define_opentype_onoffreset:nnnnn {Letters} {Unicase} {unic} {unic} {}
40 \@@_define_opentype_onoffreset:nnnnn {Letters} {Random} {rand} {rand} {}

```

2.3 Numbers

```

41 \@@_define_opentype_feature_group:n {Numbers}
42 \@@_define_opentype_feature:nnnnn {Numbers} {ResetAll} {} {}
43 {
44   +tnum,-tnum,
45   +pnum,-pnum,
46   +onum,-onum,
47   +lnum,-lnum,
48   +zero,-zero,
49   +anum,-anum,
50 }
51 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Monospaced} {tnum} {tnum} {+pnum,-pnum}
52 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Proportional} {pnum} {pnum} {+tnum,-tnum}
53 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Lowercase} {onum} {onum} {+lnum,-lnum}
54 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Uppercase} {lnum} {lnum} {+onum,-onum}
55 \@@_define_opentype_onoffreset:nnnnn {Numbers} {SlashedZero} {zero} {zero} {}
56 \aliasfontfeatureoption {Numbers} {Monospaced} {Tabular}
57 \aliasfontfeatureoption {Numbers} {Lowercase} {OldStyle}
58 \aliasfontfeatureoption {Numbers} {Uppercase} {Lining}

```

luaotload provides a custom anum feature for replacing Latin (AKA Arabic) numbers with Arabic (AKA Indic-Arabic). The same feature maps to Farsi (Persian) numbers if font language is Farsi.

```

59 \LU \@@_define_opentype_onoffreset:nnnnn {Numbers} {Arabic} {anum} {anum} {}

```

2.4 Vertical position

```

60 \@@_define_opentype_feature_group:n {VerticalPosition}
61 \@@_define_opentype_feature:nnnnn {VerticalPosition} {ResetAll} {} {}
62 {
63   +sups,-sups,
64   +subs,-subs,
65   +ordn,-ordn,
66   +numr,-numr,
67   +dnom,-dnom,
68   +sinf,-sinf,
69 }
70 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Superior} {sups} {sups} {+s}
71 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Inferior} {subs} {subs} {+s}
72 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Ordinal} {ordn} {ordn} {+s}
73 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Numerator} {numr} {numr} {+s}
74 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Denominator} {dnom} {dnom} {+s}
75 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {ScientificInferior} {sinf} {sinf} {+s}

```

2.5 Contextuals

```

76 \@@_define_opentype_feature_group:n {Contextuals}
77 \@@_define_opentype_feature:nnnnn {Contextuals} {ResetAll} {} {}
78 {
79   +cswh,-cswh,
80   +calt,-calt,
81   +init,-init,
82   +fina,-fina,
83   +falt,-falt,
84   +medi,-medi,
85 }
86 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Swash} {cswh} {cswh} {}
87 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Alternate} {calt} {calt} {}
88 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordInitial} {init} {init} {}
89 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordFinal} {fina} {fina} {}
90 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {LineFinal} {falt} {falt} {}
91 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Inner} {medi} {medi} {}

```

2.6 Diacritics

```

92 \@@_define_opentype_feature_group:n {Diacritics}
93 \@@_define_opentype_feature:nnnnn {Diacritics} {ResetAll} {} {}
94 {
95   +mark,-mark,
96   +mkmk,-mkmk,
97   +abvm,-abvm,
98   +blwm,-blwm,
99 }
100 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToBase} {mark} {mark} {}
101 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToMark} {mkmk} {mkmk} {}
102 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {AboveBase} {abvm} {abvm} {}
103 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {BelowBase} {blwm} {blwm} {}

```

2.7 Kerning

```

104 \@@_define_opentype_feature_group:n {Kerning}
105 \@@_define_opentype_feature:nnnnn {Kerning} {ResetAll} {} {}
106 {
107   +csp, -csp,
108   +kern, -kern,
109 }
110 \@@_define_opentype_onoffreset:nnnnn {Kerning} {Uppercase} {csp} {csp} {}
111 \@@_define_opentype_feature:nnnnn {Kerning} {On} {kern} {+kern} {-kern}
112 \@@_define_opentype_feature:nnnnn {Kerning} {Off} {kern} {-kern} {+kern}
113 \@@_define_opentype_feature:nnnnn {Kerning} {Reset} {} {} {+kern, -kern}

```

2.8 Fractions

```

114 \@@_define_opentype_feature_group:n {Fractions}
115 \@@_define_opentype_feature:nnnnn {Fractions} {ResetAll} {} {}
116 {
117   +frac,-frac,
118   +afrc,-afrc,
119 }

```

```

120 \@@_define_opentype_feature:nmnnn {Fractions} {On} {frac} {+frac} {}
121 \@@_define_opentype_feature:nmnnn {Fractions} {Off} {frac} {-frac} {}
122 \@@_define_opentype_feature:nmnnn {Fractions} {Reset} {} {} {+frac,-frac}
123 \@@_define_opentype_onoffreset:nmnnn {Fractions} {Alternate} {afrc} {afrc} {-frac}

124 \@@_define_opentype_feature_group:n {LocalForms}
125 \@@_define_opentype_feature:nmnnn {LocalForms} {On} {locl} {+locl} {}
126 \@@_define_opentype_feature:nmnnn {LocalForms} {Off} {locl} {-locl} {}
127 \@@_define_opentype_feature:nmnnn {LocalForms} {Reset} {} {} {+locl,-locl}

```

2.9 Style

```

128 \@@_define_opentype_feature_group:n {Style}
129 \@@_define_opentype_feature:nmnnn {Style} {ResetAll} {} {}
130 {
131   +salt,-salt,
132   +ital,-ital,
133   +ruby,-ruby,
134   +swsh,-swsh,
135   +hist,-hist,
136   +titl,-titl,
137   +hkna,-hkna,
138   +vkna,-vkna,
139   +ssty=0,-ssty=0,
140   +ssty=1,-ssty=1,
141 }
142 \@@_define_opentype_onoffreset:nmnnn {Style} {Alternate} {salt} {salt} {}
143 \@@_define_opentype_onoffreset:nmnnn {Style} {Italic} {ital} {ital} {}
144 \@@_define_opentype_onoffreset:nmnnn {Style} {Ruby} {ruby} {ruby} {}
145 \@@_define_opentype_onoffreset:nmnnn {Style} {Swash} {swsh} {swsh} {}
146 \@@_define_opentype_onoffreset:nmnnn {Style} {Cursive} {swsh} {curs} {}
147 \@@_define_opentype_onoffreset:nmnnn {Style} {Historic} {hist} {hist} {}
148 \@@_define_opentype_onoffreset:nmnnn {Style} {Titling} {titl} {titl} {}
149 \@@_define_opentype_onoffreset:nmnnn {Style} {TitlingCaps} {titl} {titl} {} % backwards
150 \@@_define_opentype_onoffreset:nmnnn {Style} {HorizontalKana} {hkna} {hkna} {+vkna,+pkna}
151 \@@_define_opentype_onoffreset:nmnnn {Style} {VerticalKana} {vkna} {vkna} {+hkna,+pkna}
152 \@@_define_opentype_onoffreset:nmnnn {Style} {ProportionalKana} {pkna} {pkna} {+vkna,+hkna}
153 \@@_define_opentype_feature:nmnnn {Style} {MathScript} {ssty} {+ssty=0} {+ssty=1}
154 \@@_define_opentype_feature:nmnnn {Style} {MathScriptScript} {ssty} {+ssty=1} {+ssty=0}
155 \@@_define_opentype_onoffreset:nmnnn {Style} {Uppercase} {case} {case} {}

```

2.10 CJK shape

```

156 \@@_define_opentype_feature_group:n {CJKShape}
157 \@@_define_opentype_feature:nmnnn {CJKShape} {ResetAll} {} {}
158 {
159   +trad,-trad,
160   +smp1,-smp1,
161   +jp78,-jp78,
162   +jp83,-jp83,
163   +jp90,-jp90,
164   +jp04,-jp04,
165   +expt,-expt,

```

```

166     +nlck,-nlck,
167 }
168 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {Traditional} {trad} {trad} {+smpl,+jp78,+jp83}
169 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {Simplified} {smpl} {smpl} {+trad,+jp78,+jp83}
170 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {JIS1978} {jp78} {jp78} {+trad,+smpl,+jp83}
171 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {JIS1983} {jp83} {jp83} {+trad,+smpl,+jp78}
172 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {JIS1990} {jp90} {jp90} {+trad,+smpl,+jp78}
173 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {JIS2004} {jp04} {jp04} {+trad,+smpl,+jp78}
174 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {Expert} {expt} {expt} {+trad,+smpl,+jp78}
175 \@@_define_opentype_onoffreset:nmnnn {CJKShape} {NLC} {nlck} {nlck} {+trad,+smpl,+jp78}

```

2.11 Character width

```

176 \@@_define_opentype_feature_group:n {CharacterWidth}
177 \@@_define_opentype_feature:nmnnn {CharacterWidth} {ResetAll} {} {}
178 {
179     +pwid,-pwid,
180     +fwid,-fwid,
181     +hwid,-hwid,
182     +twid,-twid,
183     +qwid,-qwid,
184     +palt,-palt,
185     +halt,-halt,
186 }
187 \@@_define_opentype_onoffreset:nmnnn {CharacterWidth} {Proportional} {pwid} {pwid} {+}
188 \@@_define_opentype_onoffreset:nmnnn {CharacterWidth} {Full} {fwid} {fwid} {+}
189 \@@_define_opentype_onoffreset:nmnnn {CharacterWidth} {Half} {hwid} {hwid} {+}
190 \@@_define_opentype_onoffreset:nmnnn {CharacterWidth} {Third} {twid} {twid} {+}
191 \@@_define_opentype_onoffreset:nmnnn {CharacterWidth} {Quarter} {qwid} {qwid} {+}
192 \@@_define_opentype_onoffreset:nmnnn {CharacterWidth} {AlternateProportional} {palt} {palt} {+}
193 \@@_define_opentype_onoffreset:nmnnn {CharacterWidth} {AlternateHalf} {halt} {halt} {+}

```

2.12 Vertical

According to spec vkrn must also activate vpal if available but for simplicity we don't do that here (yet?).

```

194 \@@_define_opentype_feature_group:n {Vertical}
195 \@@_define_opentype_onoffreset:nmnnn {Vertical} {RotatedGlyphs} {vrt2} {vrt2} {+vrtr,+}
196 \@@_define_opentype_onoffreset:nmnnn {Vertical} {AlternatesForRotation} {vrtr} {vrtr} {+vrt2}
197 \@@_define_opentype_onoffreset:nmnnn {Vertical} {Alternates} {vert} {vert} {+vrt2}
198 \@@_define_opentype_onoffreset:nmnnn {Vertical} {KanaAlternates} {vkna} {vkna} {+hkna}
199 \@@_define_opentype_onoffreset:nmnnn {Vertical} {Kerning} {vkrn} {vkrn} {}
200 \@@_define_opentype_onoffreset:nmnnn {Vertical} {AlternateMetrics} {valt} {valt} {+vhal,+}
201 \@@_define_opentype_onoffreset:nmnnn {Vertical} {HalfMetrics} {vhal} {vhal} {+valt,+}
202 \@@_define_opentype_onoffreset:nmnnn {Vertical} {ProportionalMetrics} {vpal} {vpal} {+valt,+}

```

3 OpenType features that need numbering

3.1 Alternate

```

203 \@@_define_opentype_feature_group:n {Alternate}

```

```

204 \keys_define:nn {fontspec-opentype}
205 {
206   Alternate .default:n = {0} ,
207   Alternate .groups:n = {opentype},
208   Alternate / unknown .code:n =
209     {
210       \clist_map_inline:nn {#1}
211       { \@@_make_OT_feature:nnn {salt}{ +salt = ##1 }{} }
212     }
213 }
214 <*LU>
215 \keys_define:nn {fontspec-opentype}
216 {
217   Alternate / Random .code:n =
218     { \@@_make_OT_feature:nnn {salt}{ +salt = random }{} } ,
219 }
220 </LU>
221 \aliasfontfeature{Alternate}{StylisticAlternates}

```

3.2 Variant / StylisticSet

```

222 \@@_define_opentype_feature_group:n {Variant}
223 \keys_define:nn {fontspec-opentype}
224 {
225   Variant .default:n = {0} ,
226   Variant .groups:n = {opentype} ,
227   Variant / unknown .code:n =
228     {
229       \clist_map_inline:nn {#1}
230       {
231         \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
232       }
233     }
234 }
235 \aliasfontfeature{Variant}{StylisticSet}

```

3.3 CharacterVariant

```

236 \@@_define_opentype_feature_group:n {CharacterVariant}
237 \use:x
238 {
239   \cs_new:Npn \exp_not:N \fontspec_parse_cv:w
240     ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
241     {
242       \@@_make_OT_feature:xxx
243       { cv \exp_not:N \two@digits {##1} }
244       { +cv \exp_not:N \two@digits {##1} = ##2 } {}
245     }
246   \keys_define:nn {fontspec-opentype}
247   {
248     CharacterVariant / unknown .code:n =
249     {

```



```

250         \clist_map_inline:nn {##1}
251         {
252             \exp_not:N \fontspec_parse_cv:w
253             #####1 \c_colon_str 0 \c_colon_str \exp_not:N \q_nil
254         }
255     }
256 }
257 }

```

Possibilities: a:0:\q_nil or a:b:0:\q_nil.

3.4 Annotation

```

258 \@@_define_opentype_feature_group:n {Annotation}
259 \keys_define:nn {fontspec-opentype}
260 {
261     Annotation .default:n = {0} ,
262     Annotation .groups:n = {opentype},
263     Annotation / unknown .code:n =
264     {
265         \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
266     }
267 }

```

3.5 Ornament

```

268 \@@_define_opentype_feature_group:n {Ornament}
269 \keys_define:nn {fontspec-opentype}
270 {
271     Ornament .default:n = {0} ,
272     Ornament .groups:n = {opentype},
273     Ornament / unknown .code:n =
274     {
275         \@@_make_OT_feature:nnn {ornm} {+ornm=#1} {}
276     }
277 }

```

4 Script and Language

4.1 Script

```

278 \keys_define:nn {fontspec-opentype}
279 {
280     Script .choice: ,
281     Script .groups:n = {opentype} ,
282 }
283 \cs_new:Nn \fontspec_new_script:nn
284 {
285     \keys_define:nn {fontspec-opentype} { Script / #1 .code:n =
286     {
287         <debug>\typeout{Trying~[Script=#1]}
288         \bool_set_false:N \l_@@_scriptlang_exist_bool
289         \clist_map_inline:nn {#2}
290         {

```

```

291         \exp_args:No \@@_check_script:NnT \l_@@_fontface_cs_tl {####1}
292         {
293 <debug>\typeout{Script~tag~found:~####1}
294         \tl_set:Nn \l_@@_script_name_tl {#1}
295         \tl_set:Nn \l_@@_script_tl {####1}
296         \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
297         \bool_set_true:N \l_@@_scriptlang_exist_bool
298         \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
299         \clist_map_break:
300     }
301 }
302
303 \bool_if:NF \l_@@_scriptlang_exist_bool
304 {
305 <debug>\typeout{Script~not~found!}
306     \bool_if:nF { \str_if_eq_p:ee {#1} {CustomDefault} }
307     {
308         \tl_clear:N \l_@@_script_name_tl
309         \@@_warning:nxx {no-script} {\l_fontspeg_fontname_tl} {#1}
310     }
311
312     \bool_if:nF
313     {
314         \str_if_eq_p:ee {#1} {Default} ||
315         \str_if_eq_p:ee {#1} {Latin} ||
316         \str_if_eq_p:ee {#1} {CustomDefault}
317     }
318     {
319         \keys_set:nn {fontspec-opentype} { Script = CustomDefault }
320     }
321 }
322 }
323 }
324 }

```

4.2 Language

```

325 \keys_define:nn {fontspec-opentype}
326 {
327     Language .choice: ,
328     Language .groups:n = {opentype} ,
329 }
330 \cs_new:Nn \fontspec_new_lang:nn
331 {
332     \keys_define:nn {fontspec-opentype} { Language / #1 .code:n =
333         {
334             \bool_set_false:N \l_@@_scriptlang_exist_bool
335             \clist_map_inline:nn {#2}
336             {
337                 \exp_args:No \@@_check_lang:NnTF \l_@@_fontface_cs_tl {####1}
338                 {
339                     \tl_set:Nn \l_@@_lang_tl {####1}

```

```

340         \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
341         \tl_gset:Nx \g_@@_single_feat_tl { language=###1 }
342         \bool_set_true:N \l_@@_scriptlang_exist_bool
343         \clist_map_break:
344     }
345 }
346 \bool_if:NF \l_@@_scriptlang_exist_bool
347 {
348     \@@_warning:nx {language-not-exist} {#1}
349     \keys_set:nn {fontspec-opentype} { Language = Default }
350 }
351 }
352 }
353 }

```

Language=Default These are special-cased to avoid the additional logic above. From memory, the OpenType default language is hardcoded to have a zero value, although this might be some X_YTeX-specific thing.

```

354 \@@_keys_define_code:nnn {fontspec-opentype} { Language / Default }
355 {
356     \tl_set:Nn \l_@@_lang_tl {dflt}
357     \int_zero:N \l_@@_language_int
358     \tl_gset:Nn \g_@@_single_feat_tl { language=dflt }
359 }

```

5 Backwards compatibility

```

360 \cs_new:Nn \@@_ot_compat:nn
361 {
362     \aliasfontfeatureoption {#1} {#2off} {No#2}
363 }
364 \@@_ot_compat:nn {Ligatures} {Rare}
365 \@@_ot_compat:nn {Ligatures} {Required}
366 \@@_ot_compat:nn {Ligatures} {Common}
367 \@@_ot_compat:nn {Ligatures} {Discretionary}
368 \@@_ot_compat:nn {Ligatures} {Contextual}
369 \@@_ot_compat:nn {Ligatures} {Historic}
370 \@@_ot_compat:nn {Numbers} {SlashedZero}
371 \@@_ot_compat:nn {Contextuals} {Swash}
372 \@@_ot_compat:nn {Contextuals} {Alternate}
373 \@@_ot_compat:nn {Contextuals} {WordInitial}
374 \@@_ot_compat:nn {Contextuals} {WordFinal}
375 \@@_ot_compat:nn {Contextuals} {LineFinal}
376 \@@_ot_compat:nn {Contextuals} {Inner}
377 \@@_ot_compat:nn {Diacritics} {MarkToBase}
378 \@@_ot_compat:nn {Diacritics} {MarkToMark}
379 \@@_ot_compat:nn {Diacritics} {AboveBase}
380 \@@_ot_compat:nn {Diacritics} {BelowBase}

```

File XV

fontspec-code-scripts.dtx

1 Font script definitions

```
1 \newfontscript{Adlam}{adlm}
2 \newfontscript{Ahom}{ahom}
3 \newfontscript{Anatolian~Hieroglyphs}{hluw}
4 \newfontscript{Arabic}{arab}
5 \newfontscript{Armenian}{armn}
6 \newfontscript{Avestan}{avst}
7 \newfontscript{Balinese}{bali}
8 \newfontscript{Bamum}{bamu}
9 \newfontscript{Bassa~Vah}{bass}
10 \newfontscript{Batak}{batk}
11 \newfontscript{Bengali}{bng2,beng}
12 \newfontscript{Bhaiksuki}{bhks}
13 \newfontscript{Bopomofo}{bopo}
14 \newfontscript{Brahmi}{brah}
15 \newfontscript{Braille}{brai}
16 \newfontscript{Buginese}{bugi}
17 \newfontscript{Buhid}{buhd}
18 \newfontscript{Byzantine~Music}{byzm}
19 \newfontscript{Canadian~Syllabics}{cans}
20 \newfontscript{Carian}{cari}
21 \newfontscript{Caucasian~Albanian}{aghb}
22 \newfontscript{Chakma}{cakm}
23 \newfontscript{Cham}{cham}
24 \newfontscript{Cherokee}{cher}
25 \newfontscript{Chorasmian}{chrs}
26 \newfontscript{CJK~Ideographic}{hani}
27 \newfontscript{Coptic}{copt}
28 \newfontscript{Cypriot~Syllabary}{cpri}
29 \newfontscript{Cypro~Minoan}{cpmn}
30 \newfontscript{Cyrillic}{cyrl}
31 \newfontscript{Default}{DFLT}
32 \newfontscript{CustomDefault}{latn,DFLT}
33 \newfontscript{Deseret}{dsrt}
34 \newfontscript{Devanagari}{dev2,deva}
35 \newfontscript{Dives~Akuru}{diak}
36 \newfontscript{Dogra}{dogr}
37 \newfontscript{Duployan}{dupl}
38 \newfontscript{Egyptian~Hieroglyphs}{egyp}
39 \newfontscript{Elbasan}{elba}
40 \newfontscript{Elymaic}{elym}
41 \newfontscript{Ethiopic}{ethi}
42 \newfontscript{Georgian}{geor}
43 \newfontscript{Glagolitic}{glag}
44 \newfontscript{Gothic}{goth}
```

```

45 \newfontscript{Grantha}{gran}
46 \newfontscript{Greek}{grek}
47 \newfontscript{Gujarati}{gjr2,gujr}
48 \newfontscript{Gunjala-Gondi}{gong}
49 \newfontscript{Gurmukhi}{gur2,guru}
50 \newfontscript{Hangul-Jamo}{jamo}
51 \newfontscript{Hangul}{hang}
52 \newfontscript{Hanifi-Rohingya}{rohng}
53 \newfontscript{Hanunoo}{hano}
54 \newfontscript{Hatran}{hatr}
55 \newfontscript{Hebrew}{hebr}
56 \newfontscript{Hiragana-and-Katakana}{kana}
57 \newfontscript{Imperial-Aramaic}{armi}
58 \newfontscript{Inscriptional-Pahlavi}{phli}
59 \newfontscript{Inscriptional-Parthian}{prti}
60 \newfontscript{Javanese}{java}
61 \newfontscript{Kaithi}{kthi}
62 \newfontscript{Kannada}{knd2,knda}
63 \newfontscript{Kawi}{kawi}
64 \newfontscript{Kayah-Li}{kali}
65 \newfontscript{Kharosthi}{khar}
66 \newfontscript{Khitan-Small-Script}{kits}
67 \newfontscript{Khmer}{khmr}
68 \newfontscript{Khojki}{khoj}
69 \newfontscript{Khudawadi}{sind}
70 \newfontscript{Lao}{lao~}
71 \newfontscript{Latin}{latn}
72 \newfontscript{Lepcha}{lepc}
73 \newfontscript{Limbu}{limb}
74 \newfontscript{Linear-A}{lina}
75 \newfontscript{Linear-B}{linb}
76 \newfontscript{Lisu}{lisu}
77 \newfontscript{Lycian}{lyci}
78 \newfontscript{Lydian}{lydi}
79 \newfontscript{Mahajani}{mahj}
80 \newfontscript{Makasar}{maka}
81 \newfontscript{Malayalam}{mlm2,mlym}
82 \newfontscript{Mandaic}{mand}
83 \newfontscript{Manichaeen}{mani}
84 \newfontscript{Marchen}{marc}
85 \newfontscript{Masaram-Gondi}{gonm}
86 \newfontscript{Math}{math}
87 \newfontscript{Medefaidrin}{medf}
88 \newfontscript{Meitei-Mayek}{mtei}
89 \newfontscript{Mende-Kikakui}{mend}
90 \newfontscript{Meroitic-Cursive}{merc}
91 \newfontscript{Meroitic-Hieroglyphs}{mero}
92 \newfontscript{Miao}{plrd}
93 \newfontscript{Modi}{modi}
94 \newfontscript{Mongolian}{mong}
95 \newfontscript{Mro}{mroo}

```

96 \newfontscript{Multani}{mult}
 97 \newfontscript{Musical~Symbols}{musc}
 98 \newfontscript{Myanmar}{mym2,mymr}
 99 \newfontscript{N'Ko}{nko~}
 100 \newfontscript{Nabataean}{nbat}
 101 \newfontscript{Nag~Mundari}{nagm}
 102 \newfontscript{Nandinagari}{nand}
 103 \newfontscript{Newa}{newa}
 104 \newfontscript{Nushu}{nshu}
 105 \newfontscript{Nyiakeng~Puachue~Hmong}{hmnpr}
 106 \newfontscript{Odia}{ory2,orya}
 107 \newfontscript{Ogham}{ogam}
 108 \newfontscript{Ol~Chiki}{olck}
 109 \newfontscript{Old~Italic}{ital}
 110 \newfontscript{Old~Hungarian}{hung}
 111 \newfontscript{Old~North~Arabian}{narb}
 112 \newfontscript{Old~Permic}{perm}
 113 \newfontscript{Old~Persian~Cuneiform}{xpeo}
 114 \newfontscript{Old~Sogdian}{sogo}
 115 \newfontscript{Old~South~Arabian}{sarb}
 116 \newfontscript{Old~Turkic}{orkh}
 117 \newfontscript{Old~Uyghur}{ougr}
 118 \newfontscript{Osage}{osge}
 119 \newfontscript{Osmanya}{osma}
 120 \newfontscript{Pahawh~Hmong}{hmng}
 121 \newfontscript{Palmyrene}{palm}
 122 \newfontscript{Pau~Cin~Hau}{pauc}
 123 \newfontscript{Phags~pa}{phag}
 124 \newfontscript{Phoenician}{phnx}
 125 \newfontscript{Psalter~Pahlavi}{phlp}
 126 \newfontscript{Rejang}{rjng}
 127 \newfontscript{Runic}{runr}
 128 \newfontscript{Samaritan}{samr}
 129 \newfontscript{Saurashtra}{saur}
 130 \newfontscript{Sharada}{shrd}
 131 \newfontscript{Shavian}{shaw}
 132 \newfontscript{Siddham}{sidd}
 133 \newfontscript{Sign~Writing}{sgnw}
 134 \newfontscript{Sinhala}{sinh}
 135 \newfontscript{Sogdian}{sogd}
 136 \newfontscript{Sora~Sompeng}{sora}
 137 \newfontscript{Sumero~Akkadian~Cuneiform}{xsux}
 138 \newfontscript{Sundanese}{sund}
 139 \newfontscript{Syloti~Nagri}{sylo}
 140 \newfontscript{Syriac}{syrz}
 141 \newfontscript{Tagalog}{tglg}
 142 \newfontscript{Tagbanwa}{tagb}
 143 \newfontscript{Tai~Le}{tale}
 144 \newfontscript{Tai~Lu}{talu}
 145 \newfontscript{Tai~Tham}{lana}
 146 \newfontscript{Tai~Viet}{tavt}

```

147 \newfontscript{Takri}{takr}
148 \newfontscript{Tamil}{tml2,taml}
149 \newfontscript{Tangsa}{tnsa}
150 \newfontscript{Tangut}{tang}
151 \newfontscript{Telugu}{tel2,telu}
152 \newfontscript{Thaana}{thaa}
153 \newfontscript{Thai}{thai}
154 \newfontscript{Tibetan}{tibt}
155 \newfontscript{Tifinagh}{tfng}
156 \newfontscript{Tirhuta}{tirh}
157 \newfontscript{Toto}{toto}
158 \newfontscript{Ugaritic~Cuneiform}{ugar}
159 \newfontscript{Vai}{vai~}
160 \newfontscript{Vithkuqi}{vith}
161 \newfontscript{Wancho}{wcho}
162 \newfontscript{Warang-Citi}{wara}
163 \newfontscript{Yezidi}{yezi}
164 \newfontscript{Yi}{yi~~}
165 \newfontscript{Zanabazar~Square}{zanb}

```

For convenience or backwards compatibility:

```

166 \newfontscript{CJK}{hani}
167 \newfontscript{Kana}{kana}
168 \newfontscript{Maths}{math}
169 \newfontscript{N'ko}{nko~}
170 \newfontscript{Oriya}{ory2,orya}

```

File XVI

fontspec-code-lang.dtx

1 Font language definitions

```
1 \newfontlanguage{Abaza}{ABA}
2 \newfontlanguage{Abkhazian}{ABK}
3 \newfontlanguage{Adyghe}{ADY}
4 \newfontlanguage{Afrikaans}{AFK}
5 \newfontlanguage{Afar}{AFR}
6 \newfontlanguage{Agaw}{AGW}
7 \newfontlanguage{Altai}{ALT}
8 \newfontlanguage{Amharic}{AMH}
9 \newfontlanguage{Arabic}{ARA}
10 \newfontlanguage{Aari}{ARI}
11 \newfontlanguage{Arakanese}{ARK}
12 \newfontlanguage{Assamese}{ASM}
13 \newfontlanguage{Athapaskan}{ATH}
14 \newfontlanguage{Avar}{AVR}
15 \newfontlanguage{Awadhi}{AWA}
16 \newfontlanguage{Aymara}{AYM}
17 \newfontlanguage{Azeri}{AZE}
18 \newfontlanguage{Badaga}{BAD}
19 \newfontlanguage{Baghelkhandi}{BAG}
20 \newfontlanguage{Balkar}{BAL}
21 \newfontlanguage{Baule}{BAU}
22 \newfontlanguage{Berber}{BBR}
23 \newfontlanguage{Bench}{BCH}
24 \newfontlanguage{Bible~Cree}{BCR}
25 \newfontlanguage{Belarussian}{BEL}
26 \newfontlanguage{Bemba}{BEM}
27 \newfontlanguage{Bengali}{BEN}
28 \newfontlanguage{Bulgarian}{BGR}
29 \newfontlanguage{Bhili}{BHI}
30 \newfontlanguage{Bhojpuri}{BHO}
31 \newfontlanguage{Bikol}{BIK}
32 \newfontlanguage{Bilen}{BIL}
33 \newfontlanguage{Blackfoot}{BKF}
34 \newfontlanguage{Balochi}{BLI}
35 \newfontlanguage{Balante}{BLN}
36 \newfontlanguage{Balti}{BLT}
37 \newfontlanguage{Bambara}{BMB}
38 \newfontlanguage{Bamileke}{BML}
39 \newfontlanguage{Breton}{BRE}
40 \newfontlanguage{Brahui}{BRH}
41 \newfontlanguage{Braj~Bhasha}{BRI}
42 \newfontlanguage{Burmese}{BRM}
43 \newfontlanguage{Bashkir}{BSH}
44 \newfontlanguage{Beti}{BTI}
```


45 \newfontlanguage{Catalan}{CAT}
46 \newfontlanguage{Cebuano}{CEB}
47 \newfontlanguage{Chechen}{CHE}
48 \newfontlanguage{Chaha~Gurage}{CHG}
49 \newfontlanguage{Chattisgarhi}{CHH}
50 \newfontlanguage{Chichewa}{CHI}
51 \newfontlanguage{Chukchi}{CHK}
52 \newfontlanguage{Chipewyan}{CHP}
53 \newfontlanguage{Cherokee}{CHR}
54 \newfontlanguage{Chuvash}{CHU}
55 \newfontlanguage{Comorian}{CMR}
56 \newfontlanguage{Coptic}{COP}
57 \newfontlanguage{Cree}{CRE}
58 \newfontlanguage{Carrier}{CRR}
59 \newfontlanguage{Crimean~Tatar}{CRT}
60 \newfontlanguage{Church~Slavonic}{CSL}
61 \newfontlanguage{Czech}{CSY}
62 \newfontlanguage{Danish}{DAN}
63 \newfontlanguage{Dargwa}{DAR}
64 \newfontlanguage{Woods~Cree}{DCR}
65 \newfontlanguage{German}{DEU}
66 \newfontlanguage{Dogri}{DGR}
67 \newfontlanguage{Divehi}{DIV}
68 \newfontlanguage{Djerma}{DJR}
69 \newfontlanguage{Dangme}{DNG}
70 \newfontlanguage{Dinka}{DNK}
71 \newfontlanguage{Dungan}{DUN}
72 \newfontlanguage{Dzongkha}{DZN}
73 \newfontlanguage{Ebira}{EBI}
74 \newfontlanguage{Eastern~Cree}{ECR}
75 \newfontlanguage{Edo}{EDO}
76 \newfontlanguage{Efik}{EFI}
77 \newfontlanguage{Greek}{ELL}
78 \newfontlanguage{English}{ENG}
79 \newfontlanguage{Erzya}{ERZ}
80 \newfontlanguage{Spanish}{ESP}
81 \newfontlanguage{Estonian}{ETI}
82 \newfontlanguage{Basque}{EUQ}
83 \newfontlanguage{Evenki}{EVK}
84 \newfontlanguage{Even}{EVN}
85 \newfontlanguage{Ewe}{EWE}
86 \newfontlanguage{French~Antillean}{FAN}
87 \newfontlanguage{Farsi}{FAR}
88 \newfontlanguage{Parsi}{FAR}
89 \newfontlanguage{Persian}{FAR}
90 \newfontlanguage{Finnish}{FIN}
91 \newfontlanguage{Fijian}{FJI}
92 \newfontlanguage{Flemish}{FLE}
93 \newfontlanguage{Forest~Nenets}{FNE}
94 \newfontlanguage{Fon}{FON}
95 \newfontlanguage{Faroese}{FOS}

96 \newfontlanguage{French}{FRA}
 97 \newfontlanguage{Frisian}{FRI}
 98 \newfontlanguage{Friulian}{FRL}
 99 \newfontlanguage{Futa}{FTA}
 100 \newfontlanguage{Fulani}{FUL}
 101 \newfontlanguage{Ga}{GAD}
 102 \newfontlanguage{Gaelic}{GAE}
 103 \newfontlanguage{Gagauz}{GAG}
 104 \newfontlanguage{Galician}{GAL}
 105 \newfontlanguage{Garshuni}{GAR}
 106 \newfontlanguage{Garhwali}{GAW}
 107 \newfontlanguage{Ge'ez}{GEZ}
 108 \newfontlanguage{Gilyak}{GIL}
 109 \newfontlanguage{Gumuz}{GMZ}
 110 \newfontlanguage{Gondi}{GON}
 111 \newfontlanguage{Greenlandic}{GRN}
 112 \newfontlanguage{Garo}{GRO}
 113 \newfontlanguage{Guarani}{GUA}
 114 \newfontlanguage{Gujarati}{GUJ}
 115 \newfontlanguage{Haitian}{HAI}
 116 \newfontlanguage{Halam}{HAL}
 117 \newfontlanguage{Harauti}{HAR}
 118 \newfontlanguage{Hausa}{HAU}
 119 \newfontlanguage{Hawaiin}{HAW}
 120 \newfontlanguage{Hammer-Banna}{HBN}
 121 \newfontlanguage{Hiligaynon}{HIL}
 122 \newfontlanguage{Hindi}{HIN}
 123 \newfontlanguage{High-Mari}{HMA}
 124 \newfontlanguage{Hindko}{HND}
 125 \newfontlanguage{Ho}{HO}
 126 \newfontlanguage{Harari}{HRI}
 127 \newfontlanguage{Croatian}{HRV}
 128 \newfontlanguage{Hungarian}{HUN}
 129 \newfontlanguage{Armenian}{HYE}
 130 \newfontlanguage{Igbo}{IBO}
 131 \newfontlanguage{Ijo}{IJO}
 132 \newfontlanguage{Ilokano}{ILO}
 133 \newfontlanguage{Indonesian}{IND}
 134 \newfontlanguage{Ingush}{ING}
 135 \newfontlanguage{Inuktitut}{INU}
 136 \newfontlanguage{Irish}{IRI}
 137 \newfontlanguage{Irish~Traditional}{IRT}
 138 \newfontlanguage{Icelandic}{ISL}
 139 \newfontlanguage{Inari~Sami}{ISM}
 140 \newfontlanguage{Italian}{ITA}
 141 \newfontlanguage{Hebrew}{IWR}
 142 \newfontlanguage{Javanese}{JAV}
 143 \newfontlanguage{Yiddish}{JII}
 144 \newfontlanguage{Japanese}{JAN}
 145 \newfontlanguage{Judezmo}{JUD}
 146 \newfontlanguage{Jula}{JUL}

147 \newfontlanguage{Kabardian}{KAB}
148 \newfontlanguage{Kachchi}{KAC}
149 \newfontlanguage{Kalenjin}{KAL}
150 \newfontlanguage{Kannada}{KAN}
151 \newfontlanguage{Karachay}{KAR}
152 \newfontlanguage{Georgian}{KAT}
153 \newfontlanguage{Kazakh}{KAZ}
154 \newfontlanguage{Kebena}{KEB}
155 \newfontlanguage{Khutsuri-Georgian}{KGE}
156 \newfontlanguage{Khakass}{KHA}
157 \newfontlanguage{Khanty-Kazim}{KHK}
158 \newfontlanguage{Khmer}{KHM}
159 \newfontlanguage{Khanty-Shurishkar}{KHS}
160 \newfontlanguage{Khanty-Vakhi}{KHV}
161 \newfontlanguage{Khowar}{KHW}
162 \newfontlanguage{Kikuyu}{KIK}
163 \newfontlanguage{Kirghiz}{KIR}
164 \newfontlanguage{Kisii}{KIS}
165 \newfontlanguage{Kokni}{KKN}
166 \newfontlanguage{Kalmyk}{KLM}
167 \newfontlanguage{Kamba}{KMB}
168 \newfontlanguage{Kumaoni}{KMN}
169 \newfontlanguage{Komo}{KMO}
170 \newfontlanguage{Komso}{KMS}
171 \newfontlanguage{Kanuri}{KNR}
172 \newfontlanguage{Kodagu}{KOD}
173 \newfontlanguage{Korean~Old~Hangul}{KOH}
174 \newfontlanguage{Konkani}{KOK}
175 \newfontlanguage{Kikongo}{KON}
176 \newfontlanguage{Komi-Permyak}{KOP}
177 \newfontlanguage{Korean}{KOR}
178 \newfontlanguage{Komi-Zyrian}{KOZ}
179 \newfontlanguage{Kpelle}{KPL}
180 \newfontlanguage{Krio}{KRI}
181 \newfontlanguage{Karakalpak}{KRK}
182 \newfontlanguage{Karelian}{KRL}
183 \newfontlanguage{Karaim}{KRM}
184 \newfontlanguage{Karen}{KRN}
185 \newfontlanguage{Koorete}{KRT}
186 \newfontlanguage{Kashmiri}{KSH}
187 \newfontlanguage{Khasi}{KSI}
188 \newfontlanguage{Kildin~Sami}{KSM}
189 \newfontlanguage{Kui}{KUI}
190 \newfontlanguage{Kulvi}{KUL}
191 \newfontlanguage{Kumyk}{KUM}
192 \newfontlanguage{Kurdish}{KUR}
193 \newfontlanguage{Kurukh}{KUU}
194 \newfontlanguage{Kuy}{KUY}
195 \newfontlanguage{Koryak}{KYK}
196 \newfontlanguage{Ladin}{LAD}
197 \newfontlanguage{Lahuli}{LAH}

198 \newfontlanguage{Lak}{LAK}
 199 \newfontlanguage{Lambani}{LAM}
 200 \newfontlanguage{Lao}{LAO}
 201 \newfontlanguage{Latin}{LAT}
 202 \newfontlanguage{Laz}{LAZ}
 203 \newfontlanguage{L-Cree}{LCR}
 204 \newfontlanguage{Ladakhi}{LDK}
 205 \newfontlanguage{Lezgi}{LEZ}
 206 \newfontlanguage{Lingala}{LIN}
 207 \newfontlanguage{Low-Mari}{LMA}
 208 \newfontlanguage{Limbu}{LMB}
 209 \newfontlanguage{Lomwe}{LMW}
 210 \newfontlanguage{Lower-Sorbian}{LSB}
 211 \newfontlanguage{Lule-Sami}{LSM}
 212 \newfontlanguage{Lithuanian}{LTH}
 213 \newfontlanguage{Luba}{LUB}
 214 \newfontlanguage{Luganda}{LUG}
 215 \newfontlanguage{Luhya}{LUH}
 216 \newfontlanguage{Luo}{LUO}
 217 \newfontlanguage{Latvian}{LVI}
 218 \newfontlanguage{Majang}{MAJ}
 219 \newfontlanguage{Makua}{MAK}
 220 \newfontlanguage{Malayalam-Traditional}{MAL}
 221 \newfontlanguage{Mansi}{MAN}
 222 \newfontlanguage{Marathi}{MAR}
 223 \newfontlanguage{Marwari}{MAW}
 224 \newfontlanguage{Mbundu}{MBN}
 225 \newfontlanguage{Manchu}{MCH}
 226 \newfontlanguage{Moose-Cree}{MCR}
 227 \newfontlanguage{Mende}{MDE}
 228 \newfontlanguage{Me'en}{MEN}
 229 \newfontlanguage{Mizo}{MIZ}
 230 \newfontlanguage{Macedonian}{MKD}
 231 \newfontlanguage{Male}{MLE}
 232 \newfontlanguage{Malagasy}{MLG}
 233 \newfontlanguage{Malinke}{MLN}
 234 \newfontlanguage{Malayalam-Reformed}{MLR}
 235 \newfontlanguage{Malay}{MLY}
 236 \newfontlanguage{Mandinka}{MND}
 237 \newfontlanguage{Mongolian}{MNG}
 238 \newfontlanguage{Manipuri}{MNI}
 239 \newfontlanguage{Maninka}{MNK}
 240 \newfontlanguage{Manx-Gaelic}{MNX}
 241 \newfontlanguage{Moksha}{MOK}
 242 \newfontlanguage{Moldavian}{MOL}
 243 \newfontlanguage{Mon}{MON}
 244 \newfontlanguage{Moroccan}{MOR}
 245 \newfontlanguage{Maori}{MRI}
 246 \newfontlanguage{Maithili}{MTH}
 247 \newfontlanguage{Maltese}{MTS}
 248 \newfontlanguage{Mundari}{MUN}

249 \newfontlanguage{Naga-Assamese}{NAG}
250 \newfontlanguage{Nanai}{NAN}
251 \newfontlanguage{Naskapi}{NAS}
252 \newfontlanguage{N-Cree}{NCR}
253 \newfontlanguage{Ndebele}{NDB}
254 \newfontlanguage{Ndonga}{NDG}
255 \newfontlanguage{Nepali}{NEP}
256 \newfontlanguage{Newari}{NEW}
257 \newfontlanguage{Nagari}{NGR}
258 \newfontlanguage{Norway~House~Cree}{NHC}
259 \newfontlanguage{Nisi}{NIS}
260 \newfontlanguage{Niuean}{NIU}
261 \newfontlanguage{Nkole}{NKL}
262 \newfontlanguage{N'ko}{NKO}
263 \newfontlanguage{Dutch}{NLD}
264 \newfontlanguage{Nogai}{NOG}
265 \newfontlanguage{Norwegian}{NOR}
266 \newfontlanguage{Northern-Sami}{NSM}
267 \newfontlanguage{Northern-Tai}{NTA}
268 \newfontlanguage{Esperanto}{NTO}
269 \newfontlanguage{Nynorsk}{NYN}
270 \newfontlanguage{Oji-Cree}{OCR}
271 \newfontlanguage{Ojibway}{OBJ}
272 \newfontlanguage{Oriya}{ORI}
273 \newfontlanguage{Oromo}{ORO}
274 \newfontlanguage{Ossetian}{OSS}
275 \newfontlanguage{Palestinian~Aramaic}{PAA}
276 \newfontlanguage{Pali}{PAL}
277 \newfontlanguage{Punjabi}{PAN}
278 \newfontlanguage{Palpa}{PAP}
279 \newfontlanguage{Pashto}{PAS}
280 \newfontlanguage{Polytonic~Greek}{PGR}
281 \newfontlanguage{Pilipino}{PIL}
282 \newfontlanguage{Palaung}{PLG}
283 \newfontlanguage{Polish}{PLK}
284 \newfontlanguage{Provencal}{PRO}
285 \newfontlanguage{Portuguese}{PTG}
286 \newfontlanguage{Chin}{QIN}
287 \newfontlanguage{Rajasthani}{RAJ}
288 \newfontlanguage{R-Cree}{RCR}
289 \newfontlanguage{Russian~Buriat}{RBU}
290 \newfontlanguage{Riang}{RIA}
291 \newfontlanguage{Rhaeto-Romanic}{RMS}
292 \newfontlanguage{Romanian}{ROM}
293 \newfontlanguage{Romany}{ROY}
294 \newfontlanguage{Rusyn}{RSY}
295 \newfontlanguage{Ruanda}{RUA}
296 \newfontlanguage{Russian}{RUS}
297 \newfontlanguage{Sadri}{SAD}
298 \newfontlanguage{Sanskrit}{SAN}
299 \newfontlanguage{Santali}{SAT}

300 \newfontlanguage{Sayisi}{SAY}
301 \newfontlanguage{Sekota}{SEK}
302 \newfontlanguage{Selkup}{SEL}
303 \newfontlanguage{Sango}{SGO}
304 \newfontlanguage{Shan}{SHN}
305 \newfontlanguage{Sibe}{SIB}
306 \newfontlanguage{Sidamo}{SID}
307 \newfontlanguage{Silte~Gurage}{SIG}
308 \newfontlanguage{Skolt~Sami}{SKS}
309 \newfontlanguage{Slovak}{SKY}
310 \newfontlanguage{Slavey}{SLA}
311 \newfontlanguage{Slovenian}{SLV}
312 \newfontlanguage{Somali}{SML}
313 \newfontlanguage{Samoan}{SMO}
314 \newfontlanguage{Sena}{SNA}
315 \newfontlanguage{Sindhi}{SND}
316 \newfontlanguage{Sinhalese}{SNH}
317 \newfontlanguage{Soninke}{SNK}
318 \newfontlanguage{Sodo~Gurage}{SOG}
319 \newfontlanguage{Sotho}{SOT}
320 \newfontlanguage{Albanian}{SQI}
321 \newfontlanguage{Serbian}{SRB}
322 \newfontlanguage{Saraiki}{SRK}
323 \newfontlanguage{Serer}{SRR}
324 \newfontlanguage{South~Slavey}{SSL}
325 \newfontlanguage{Southern~Sami}{SSM}
326 \newfontlanguage{Suri}{SUR}
327 \newfontlanguage{Svan}{SVA}
328 \newfontlanguage{Swedish}{SVE}
329 \newfontlanguage{Swadaya~Aramaic}{SWA}
330 \newfontlanguage{Swahili}{SWK}
331 \newfontlanguage{Swazi}{SWZ}
332 \newfontlanguage{Sutu}{SXT}
333 \newfontlanguage{Syriac}{SYR}
334 \newfontlanguage{Tabasaran}{TAB}
335 \newfontlanguage{Tajiki}{TAJ}
336 \newfontlanguage{Tamil}{TAM}
337 \newfontlanguage{Tatar}{TAT}
338 \newfontlanguage{TH~Cree}{TCR}
339 \newfontlanguage{Telugu}{TEL}
340 \newfontlanguage{Tongan}{TGN}
341 \newfontlanguage{Tigre}{TGR}
342 \newfontlanguage{Tigrinya}{TGY}
343 \newfontlanguage{Thai}{THA}
344 \newfontlanguage{Tahitian}{THT}
345 \newfontlanguage{Tibetan}{TIB}
346 \newfontlanguage{Turkish}{TRK, TUR}
347 \newfontlanguage{Turkmen}{TKM}
348 \newfontlanguage{Temne}{TMN}
349 \newfontlanguage{Tswana}{TNA}
350 \newfontlanguage{Tundra~Nenets}{TNE}

```

351 \newfontlanguage{Tonga}{TNG}
352 \newfontlanguage{Todo}{TOD}
353 \newfontlanguage{Tsonga}{TSG}
354 \newfontlanguage{Turoyo~Aramaic}{TUA}
355 \newfontlanguage{Tulu}{TUL}
356 \newfontlanguage{Tuvin}{TUV}
357 \newfontlanguage{Twi}{TWI}
358 \newfontlanguage{Udmurt}{UDM}
359 \newfontlanguage{Ukrainian}{UKR}
360 \newfontlanguage{Urdu}{URD}
361 \newfontlanguage{Upper~Sorbian}{USB}
362 \newfontlanguage{Uyghur}{UYG}
363 \newfontlanguage{Uzbek}{UZB}
364 \newfontlanguage{Venda}{VEN}
365 \newfontlanguage{Vietnamese}{VIT}
366 \newfontlanguage{Wa}{WA}
367 \newfontlanguage{Wagdi}{WAG}
368 \newfontlanguage{West-Cree}{WCR}
369 \newfontlanguage{Welsh}{WEL}
370 \newfontlanguage{Wolof}{WLF}
371 \newfontlanguage{Tai~Lue}{XBD}
372 \newfontlanguage{Xhosa}{XHS}
373 \newfontlanguage{Yakut}{YAK}
374 \newfontlanguage{Yoruba}{YBA}
375 \newfontlanguage{Y-Cree}{YCR}
376 \newfontlanguage{Yi-Classic}{YIC}
377 \newfontlanguage{Yi-Modern}{YIM}
378 \newfontlanguage{Chinese~Hong~Kong}{ZHH}
379 \newfontlanguage{Chinese~Phonetic}{ZHP}
380 \newfontlanguage{Chinese~Simplified}{ZHS}
381 \newfontlanguage{Chinese~Traditional}{ZHT}
382 \newfontlanguage{Zande}{ZND}
383 \newfontlanguage{Zulu}{ZUL}

```

File XVII

fontspec-code-feat-aat.dtx

1 AAT feature definitions

These are only defined for X₃TeX.

1.1 Ligatures

```
1 \@@_define_aat_feature_group:n {Ligatures}
2 \@@_define_aat_feature:nmmm {Ligatures} {Required} {1} {0}
3 \@@_define_aat_feature:nmmm {Ligatures} {NoRequired} {1} {1}
4 \@@_define_aat_feature:nmmm {Ligatures} {Common} {1} {2}
5 \@@_define_aat_feature:nmmm {Ligatures} {NoCommon} {1} {3}
6 \@@_define_aat_feature:nmmm {Ligatures} {Rare} {1} {4}
7 \@@_define_aat_feature:nmmm {Ligatures} {NoRare} {1} {5}
8 \@@_define_aat_feature:nmmm {Ligatures} {Discretionary} {1} {4}
9 \@@_define_aat_feature:nmmm {Ligatures} {NoDiscretionary} {1} {5}
10 \@@_define_aat_feature:nmmm {Ligatures} {Logos} {1} {6}
11 \@@_define_aat_feature:nmmm {Ligatures} {NoLogos} {1} {7}
12 \@@_define_aat_feature:nmmm {Ligatures} {Rebus} {1} {8}
13 \@@_define_aat_feature:nmmm {Ligatures} {NoRebus} {1} {9}
14 \@@_define_aat_feature:nmmm {Ligatures} {Diphthong} {1} {10}
15 \@@_define_aat_feature:nmmm {Ligatures} {NoDiphthong} {1} {11}
16 \@@_define_aat_feature:nmmm {Ligatures} {Squared} {1} {12}
17 \@@_define_aat_feature:nmmm {Ligatures} {NoSquared} {1} {13}
18 \@@_define_aat_feature:nmmm {Ligatures} {AbbrevSquared} {1} {14}
19 \@@_define_aat_feature:nmmm {Ligatures} {NoAbbrevSquared} {1} {15}
20 \@@_define_aat_feature:nmmm {Ligatures} {Icelandic} {1} {32}
21 \@@_define_aat_feature:nmmm {Ligatures} {NoIcelandic} {1} {33}
```

Emulate CM extra ligatures.

```
22 \keys_define:nn {fontspec-aat}
23 {
24   Ligatures / TeX .code:n =
25   {
26     \tl_set:Nn \l_@@_mapping_tl { tex-text }
27   }
28 }
```

1.2 Letters

```
29 \@@_define_aat_feature_group:n {Letters}
30 \@@_define_aat_feature:nmmm {Letters} {Normal} {3} {0}
31 \@@_define_aat_feature:nmmm {Letters} {Uppercase} {3} {1}
32 \@@_define_aat_feature:nmmm {Letters} {Lowercase} {3} {2}
33 \@@_define_aat_feature:nmmm {Letters} {SmallCaps} {3} {3}
34 \@@_define_aat_feature:nmmm {Letters} {InitialCaps} {3} {4}
```


1.3 Numbers

These were originally separated into `NumberCase` and `NumberSpacing` following AAT, but it makes more sense to combine them.

Both naming conventions are offered to select the number case.

```
35 \@@_define_aat_feature_group:n {Numbers}
36 \@@_define_aat_feature:nmmn {Numbers} {Monospaced} {6} {0}
37 \@@_define_aat_feature:nmmn {Numbers} {Proportional} {6} {1}
38 \@@_define_aat_feature:nmmn {Numbers} {Lowercase} {21} {0}
39 \@@_define_aat_feature:nmmn {Numbers} {OldStyle} {21} {0}
40 \@@_define_aat_feature:nmmn {Numbers} {Uppercase} {21} {1}
41 \@@_define_aat_feature:nmmn {Numbers} {Lining} {21} {1}
42 \@@_define_aat_feature:nmmn {Numbers} {SlashedZero} {14} {5}
43 \@@_define_aat_feature:nmmn {Numbers} {NoSlashedZero} {14} {4}
```

1.4 Contextuals

```
44 \@@_define_aat_feature_group:n {Contextuals}
45 \@@_define_aat_feature:nmmn {Contextuals} {WordInitial} {8} {0}
46 \@@_define_aat_feature:nmmn {Contextuals} {NoWordInitial} {8} {1}
47 \@@_define_aat_feature:nmmn {Contextuals} {WordFinal} {8} {2}
48 \@@_define_aat_feature:nmmn {Contextuals} {NoWordFinal} {8} {3}
49 \@@_define_aat_feature:nmmn {Contextuals} {LineInitial} {8} {4}
50 \@@_define_aat_feature:nmmn {Contextuals} {NoLineInitial} {8} {5}
51 \@@_define_aat_feature:nmmn {Contextuals} {LineFinal} {8} {6}
52 \@@_define_aat_feature:nmmn {Contextuals} {NoLineFinal} {8} {7}
53 \@@_define_aat_feature:nmmn {Contextuals} {Inner} {8} {8}
54 \@@_define_aat_feature:nmmn {Contextuals} {NoInner} {8} {9}
```

1.5 Diacritics

```
55 \@@_define_aat_feature_group:n {Diacritics}
56 \@@_define_aat_feature:nmmn {Diacritics} {Show} {9} {0}
57 \@@_define_aat_feature:nmmn {Diacritics} {Hide} {9} {1}
58 \@@_define_aat_feature:nmmn {Diacritics} {Decompose} {9} {2}
```

1.6 Vertical position

```
59 \@@_define_aat_feature_group:n {VerticalPosition}
60 \@@_define_aat_feature:nmmn {VerticalPosition} {Normal} {10} {0}
61 \@@_define_aat_feature:nmmn {VerticalPosition} {Superior} {10} {1}
62 \@@_define_aat_feature:nmmn {VerticalPosition} {Inferior} {10} {2}
63 \@@_define_aat_feature:nmmn {VerticalPosition} {Ordinal} {10} {3}
```

1.7 Fractions

```
64 \@@_define_aat_feature_group:n {Fractions}
65 \@@_define_aat_feature:nmmn {Fractions} {On} {11} {1}
66 \@@_define_aat_feature:nmmn {Fractions} {Off} {11} {0}
67 \@@_define_aat_feature:nmmn {Fractions} {Diagonal} {11} {2}
```

1.8 Alternate

```
68 \@@_define_aat_feature_group:n { Alternate }
```

```

69 \keys_define:nn {fontspec-aat}
70 {
71   Alternate .default:n = {0} ,
72   Alternate / unknown .code:n =
73     {
74       \clist_map_inline:nn {#1}
75         {
76           \@@_make_AAT_feature:nn {17}{##1}
77         }
78     }
79 }

```

1.9 Variant / StylisticSet

```

80 \@@_define_aat_feature_group:n {Variant}
81 \keys_define:nn {fontspec-aat}
82 {
83   Variant .default:n = {0} ,
84   Variant / unknown .code:n =
85     {
86       \clist_map_inline:nn {#1}
87         { \@@_make_AAT_feature:nn {18}{##1} }
88     }
89 }
90 \aliasfontfeature{Variant}{StylisticSet}
91 \@@_define_aat_feature_group:n {Vertical}
92 \keys_define:nn {fontspec-aat}
93 {
94   Vertical .choice: ,
95   Vertical / RotatedGlyphs .code:n =
96     {
97       \__fontspec_update_featstr:n {vertical}
98     }
99 }

```

1.10 Style

```

100 \@@_define_aat_feature_group:n {Style}
101 \@@_define_aat_feature:nnnn {Style} {Italic} {32} {2}
102 \@@_define_aat_feature:nnnn {Style} {Ruby} {28} {2}
103 \@@_define_aat_feature:nnnn {Style} {Display} {19} {1}
104 \@@_define_aat_feature:nnnn {Style} {Engraved} {19} {2}
105 \@@_define_aat_feature:nnnn {Style} {Titling} {19} {4}
106 \@@_define_aat_feature:nnnn {Style} {TitlingCaps} {19} {4} % backwards compat
107 \@@_define_aat_feature:nnnn {Style} {TallCaps} {19} {5}

```

1.11 CJK shape

```

108 \@@_define_aat_feature_group:n {CJKShape}
109 \@@_define_aat_feature:nnnn {CJKShape} {Traditional} {20} {0}
110 \@@_define_aat_feature:nnnn {CJKShape} {Simplified} {20} {1}
111 \@@_define_aat_feature:nnnn {CJKShape} {JIS1978} {20} {2}
112 \@@_define_aat_feature:nnnn {CJKShape} {JIS1983} {20} {3}

```

`113 \@@_define_aat_feature:nmmm {CJKShape} {JIS1990} {20} {4}`
`114 \@@_define_aat_feature:nmmm {CJKShape} {Expert} {20} {10}`
`115 \@@_define_aat_feature:nmmm {CJKShape} {NLC} {20} {13}`

1.12 Character width

`116 \@@_define_aat_feature_group:n {CharacterWidth}`
`117 \@@_define_aat_feature:nmmm {CharacterWidth} {Proportional} {22} {0}`
`118 \@@_define_aat_feature:nmmm {CharacterWidth} {Full} {22} {1}`
`119 \@@_define_aat_feature:nmmm {CharacterWidth} {Half} {22} {2}`
`120 \@@_define_aat_feature:nmmm {CharacterWidth} {Third} {22} {3}`
`121 \@@_define_aat_feature:nmmm {CharacterWidth} {Quarter} {22} {4}`
`122 \@@_define_aat_feature:nmmm {CharacterWidth} {AlternateProportional} {22} {5}`
`123 \@@_define_aat_feature:nmmm {CharacterWidth} {AlternateHalf} {22} {6}`
`124 \@@_define_aat_feature:nmmm {CharacterWidth} {Default} {22} {7}`

1.13 Annotation

`125 \@@_define_aat_feature_group:n {Annotation}`
`126 \@@_define_aat_feature:nmmm {Annotation} {Off} {24} {0}`
`127 \@@_define_aat_feature:nmmm {Annotation} {Box} {24} {1}`
`128 \@@_define_aat_feature:nmmm {Annotation} {RoundedBox} {24} {2}`
`129 \@@_define_aat_feature:nmmm {Annotation} {Circle} {24} {3}`
`130 \@@_define_aat_feature:nmmm {Annotation} {BlackCircle} {24} {4}`
`131 \@@_define_aat_feature:nmmm {Annotation} {Parenthesis} {24} {5}`
`132 \@@_define_aat_feature:nmmm {Annotation} {Period} {24} {6}`
`133 \@@_define_aat_feature:nmmm {Annotation} {RomanNumerals} {24} {7}`
`134 \@@_define_aat_feature:nmmm {Annotation} {Diamond} {24} {8}`
`135 \@@_define_aat_feature:nmmm {Annotation} {BlackSquare} {24} {9}`
`136 \@@_define_aat_feature:nmmm {Annotation} {BlackRoundSquare} {24} {10}`
`137 \@@_define_aat_feature:nmmm {Annotation} {DoubleCircle} {24} {11}`

File XVIII

fontspec-code-enc.dtx

1 Extended font encodings

`\EncodingCommand`

```
1 \DeclareDocumentCommand \EncodingCommand { m O{} O{} m }
2 {
3   \bool_if:NF \l_@@_defining_encoding_bool
4     { \@@_error:nn {only-inside-encdef} \EncodingCommand }
5   \DeclareTextCommand{#1}{\UnicodeEncodingName}{#2}[#3]{#4}
6 }
```

(End of definition for \EncodingCommand. This function is documented on page ??.)

`\EncodingAccent`

```
7 \DeclareDocumentCommand \EncodingAccent {mm}
8 {
9   \bool_if:NF \l_@@_defining_encoding_bool
10    { \@@_error:nn {only-inside-encdef} \EncodingAccent }
11   \DeclareTextCommand{#1}{\UnicodeEncodingName}{\add@unicode@accent{#2}}
12 }
```

(End of definition for \EncodingAccent. This function is documented on page ??.)

`\EncodingSymbol`

```
13 \DeclareDocumentCommand \EncodingSymbol {mm}
14 {
15   \bool_if:NF \l_@@_defining_encoding_bool
16     { \@@_error:nn {only-inside-encdef} \EncodingSymbol }
17   \DeclareTextSymbol{#1}{\UnicodeEncodingName}{#2}
18 }
```

(End of definition for \EncodingSymbol. This function is documented on page ??.)

`\EncodingComposite`

```
19 \DeclareDocumentCommand \EncodingComposite {mmm}
20 {
21   \bool_if:NF \l_@@_defining_encoding_bool
22     { \@@_error:nn {only-inside-encdef} \EncodingComposite }
23   \DeclareTextComposite{#1}{\UnicodeEncodingName}{#2}{#3}
24 }
```

(End of definition for \EncodingComposite. This function is documented on page ??.)

`\EncodingCompositeCommand`

```
25 \DeclareDocumentCommand \EncodingCompositeCommand {mmm}
26 {
27   \bool_if:NF \l_@@_defining_encoding_bool
28     { \@@_error:nn {only-inside-encdef} \EncodingCompositeCommand }
29   \DeclareTextCompositeCommand{#1}{\UnicodeEncodingName}{#2}{#3}
30 }
```

(End of definition for `\EncodingCompositeCommand`. This function is documented on page ??.)

`\DeclareUnicodeEncoding`

```
31 \DeclareDocumentCommand \DeclareUnicodeEncoding {mm}
32 {
33   \DeclareFontEncoding{#1}{}{}
34   \DeclareFontSubstitution{#1}{lmr}{m}{n}
35   \DeclareFontFamily{#1}{lmr}{}
36
37   \DeclareFontShape{#1}{lmr}{m}{n}
38     {<->\UnicodeFontFile{lmroman10-regular}{\UnicodeFontTeXLigatures}}{}
39   \DeclareFontShape{#1}{lmr}{m}{it}
40     {<->\UnicodeFontFile{lmroman10-italic}{\UnicodeFontTeXLigatures}}{}
41   \DeclareFontShape{#1}{lmr}{m}{sc}
42     {<->\UnicodeFontFile{lmromancaps10-regular}{\UnicodeFontTeXLigatures}}{}
43   \DeclareFontShape{#1}{lmr}{bx}{n}
44     {<->\UnicodeFontFile{lmroman10-bold}{\UnicodeFontTeXLigatures}}{}
45   \DeclareFontShape{#1}{lmr}{bx}{it}
46     {<->\UnicodeFontFile{lmroman10-bolditalic}{\UnicodeFontTeXLigatures}}{}
47
48   \tl_set_eq:NN \l_@@_prev_unicode_name_tl \UnicodeEncodingName
49   \tl_set:Nn \UnicodeEncodingName {#1}
50   \bool_set_true:N \l_@@_defining_encoding_bool
51   #2
52   \bool_set_false:N \l_@@_defining_encoding_bool
53   \tl_set_eq:NN \UnicodeEncodingName \l_@@_prev_unicode_name_tl
54 }
```

(End of definition for `\DeclareUnicodeEncoding`. This function is documented on page ??.)

`\UndeclareSymbol` Synonyms for each other but all included for completeness.

`\UndeclareAccent`
`\UndeclareCommand`

```
55 \DeclareDocumentCommand \UndeclareSymbol {m}
56 {
57   \bool_if:NF \l_@@_defining_encoding_bool
58     { \@@_error:nn {only-inside-encdef} \UndeclareSymbol }
59   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
60 }
61 \DeclareDocumentCommand \UndeclareAccent {m}
62 {
63   \bool_if:NF \l_@@_defining_encoding_bool
64     { \@@_error:nn {only-inside-encdef} \UndeclareAccent }
65   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
66 }
67 \DeclareDocumentCommand \UndeclareCommand {m}
68 {
69   \bool_if:NF \l_@@_defining_encoding_bool
70     { \@@_error:nn {only-inside-encdef} \UndeclareCommand }
71   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
72 }
```

(End of definition for `\UndeclareSymbol`, `\UndeclareAccent`, and `\UndeclareCommand`. These functions are documented on page ??.)

`\UndeclareComposite`

```
73 \DeclareDocumentCommand \UndeclareComposite {mm}
74 {
75   \bool_if:NF \l_@@_defining_encoding_bool
76   { \@@_error:nn {only-inside-encdef} \UndeclareComposite }
77   \cs_undefine:c
78   { \c_backslash_str \UnicodeEncodingName \token_to_str:N #1 - \tl_to_str:n {#2} }
79 }
```

(End of definition for `\UndeclareComposite`. This function is documented on page ??.)

File XIX

fontspec-code-math.dtx

1 Selecting maths fonts

Here, the fonts used in math mode are redefined to correspond to the default roman, sans serif and typewriter fonts. Unfortunately, you can only define maths fonts in the preamble, otherwise I'd run this code whenever `\setmainfont` and friends was run.

`\fontspec_setup_maths:` Everything here is performed `\AtBeginDocument` in order to overwrite euler's attempt. This means fontspec must be loaded *after* euler. We set up a conditional to return an error if this rule is violated.

Since every maths setup is slightly different, we also take different paths for defining various math glyphs depending which maths font package has been loaded.

```
1 \ifpackageloaded{euler}
2   { \bool_gset_true:N \g_@@_pkg_euler_loaded_bool }
3   { \bool_gset_false:N \g_@@_pkg_euler_loaded_bool }
4 \cs_new:Nn \fontspec_setup_maths:
5   {
6     \ifpackageloaded{euler}
7       {
8         \bool_if:NTF \g_@@_pkg_euler_loaded_bool
9           { \bool_gset_true:N \g_@@_math_euler_bool }
10          { \@@_error:n {euler-too-late} }
11        }
12      {}
13 \ifpackageloaded{lucbmath}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}
14 \ifpackageloaded{lucidabr}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}
15 \ifpackageloaded{lucimatx}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}

```

Knuth's CM fonts are all squashed together, combining letters, accents, text symbols and maths symbols all in the one font, `cmr`, plus other things in other fonts. Because we are changing the roman font in the document, we need to redefine all of the maths glyphs in \LaTeX 's `operators` maths font to still go back to the legacy `cmr` font for all these random glyphs, unless a separate maths font package has been loaded instead.

In every case, the maths accents are always taken from the `operators` font, which is generally the main text font. (Actually, there is a `\hat` accent in `EulerFraktur`, but it's *ugly*. So I ignore it. Sorry if this causes inconvenience.)

```
16 \DeclareSymbolFont{legacymaths}{OT1}{cmr}{m}{n}
17 \SetSymbolFont{legacymaths}{bold}{OT1}{cmr}{bx}{n}
18 \DeclareMathAccent{\acute}    {\mathalpha}{legacymaths}{19}
19 \DeclareMathAccent{\grave}   {\mathalpha}{legacymaths}{18}
20 \DeclareMathAccent{\ddot}    {\mathalpha}{legacymaths}{127}
21 \DeclareMathAccent{\tilde}   {\mathalpha}{legacymaths}{126}
22 \DeclareMathAccent{\bar}     {\mathalpha}{legacymaths}{22}
23 \DeclareMathAccent{\breve}   {\mathalpha}{legacymaths}{21}
24 \DeclareMathAccent{\check}   {\mathalpha}{legacymaths}{20}
25 \DeclareMathAccent{\hat}     {\mathalpha}{legacymaths}{94} % too bad, euler

```

```

26 \DeclareMathAccent{\dot}      {\mathalpha}{legacymaths}{95}
27 \DeclareMathAccent{\mathring}{\mathalpha}{legacymaths}{23}

```

`\colon`: what's going on? Okay, so `:` and `\colon` in maths mode are defined in a few places, so I need to work out what does what. Respectively, we have:

```

% % fontmath.ltx:
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}
% \DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
%
% % amsmath.sty:
% \renewcommand{\colon}{\nobreak\mskip2mu\mathpunct{}\nonscript
% \mkern-\thinmuskip{:}\mskip6mu\relax}
%
% % euler.sty:
% \DeclareMathSymbol{:}{\mathrel}{EulerFraktur}{"3A}
%
% % lucbmath.sty:
% \DeclareMathSymbol{\@tempb}{\mathpunct}{operators}{58}
% \ifx\colon\@tempb
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{58}
% \fi
% \DeclareMathSymbol{:}{\mathrel}{operators}{58}

```

(3A₁₆ = 58₁₀) So I think, based on this summary, that it is fair to tell fontspec to 'replace' the operators font with legacymaths for this symbol, except when amsmath is loaded since we want to keep its definition.

```

28 \group_begin:
29 \mathchardef\@tempa="603A \relax
30 \ifx\colon\@tempa
31 \DeclareMathSymbol{\colon}{\mathpunct}{legacymaths}{58}
32 \fi
33 \group_end:

```

The following symbols are only defined specifically in euler, so skip them if that package is loaded.

```

34 \bool_if:NF \g_@@_math_euler_bool
35 {
36 \DeclareMathSymbol{!}{\mathclose}{legacymaths}{33}
37 \DeclareMathSymbol{:}{\mathrel}{legacymaths}{58}
38 \DeclareMathSymbol{;}{\mathpunct}{legacymaths}{59}
39 \DeclareMathSymbol{?}{\mathclose}{legacymaths}{63}

```

And these ones are defined both in euler and lucbmath, so we only need to run this code if no extra maths package has been loaded.

```

40 \bool_if:NF \g_@@_math_lucida_bool
41 {
42 \DeclareMathSymbol{0}{\mathalpha}{legacymaths}{`0}
43 \DeclareMathSymbol{1}{\mathalpha}{legacymaths}{`1}
44 \DeclareMathSymbol{2}{\mathalpha}{legacymaths}{`2}

```



```

45 \DeclareMathSymbol{3}{\mathalpha}{legacymaths}{`3}
46 \DeclareMathSymbol{4}{\mathalpha}{legacymaths}{`4}
47 \DeclareMathSymbol{5}{\mathalpha}{legacymaths}{`5}
48 \DeclareMathSymbol{6}{\mathalpha}{legacymaths}{`6}
49 \DeclareMathSymbol{7}{\mathalpha}{legacymaths}{`7}
50 \DeclareMathSymbol{8}{\mathalpha}{legacymaths}{`8}
51 \DeclareMathSymbol{9}{\mathalpha}{legacymaths}{`9}
52 \DeclareMathSymbol{\Gamma}{\mathalpha}{legacymaths}{0}
53 \DeclareMathSymbol{\Delta}{\mathalpha}{legacymaths}{1}
54 \DeclareMathSymbol{\Theta}{\mathalpha}{legacymaths}{2}
55 \DeclareMathSymbol{\Lambda}{\mathalpha}{legacymaths}{3}
56 \DeclareMathSymbol{\Xi}{\mathalpha}{legacymaths}{4}
57 \DeclareMathSymbol{\Pi}{\mathalpha}{legacymaths}{5}
58 \DeclareMathSymbol{\Sigma}{\mathalpha}{legacymaths}{6}
59 \DeclareMathSymbol{\Upsilon}{\mathalpha}{legacymaths}{7}
60 \DeclareMathSymbol{\Phi}{\mathalpha}{legacymaths}{8}
61 \DeclareMathSymbol{\Psi}{\mathalpha}{legacymaths}{9}
62 \DeclareMathSymbol{\Omega}{\mathalpha}{legacymaths}{10}
63 \DeclareMathSymbol{+}{\mathbin}{legacymaths}{43}
64 \DeclareMathSymbol{=}{\mathrel}{legacymaths}{61}
65 \DeclareMathDelimiter{()}{\mathopen}{legacymaths}{40}{largesymbols}{0}
66 \DeclareMathDelimiter{)}{\mathclose}{legacymaths}{41}{largesymbols}{1}
67 \DeclareMathDelimiter{[]}{\mathopen}{legacymaths}{91}{largesymbols}{2}
68 \DeclareMathDelimiter{[]}{\mathclose}{legacymaths}{93}{largesymbols}{3}
69 \DeclareMathDelimiter{/}{\mathord}{legacymaths}{47}{largesymbols}{14}
70 \DeclareMathSymbol{\mathdollar}{\mathord}{legacymaths}{36}
71 \renewcommand{\hbar}{\mathchar"AF\mkern-9mu h}}% TODO: test with other fonts
72 }
73 }

```

Finally, we change the font definitions for `\mathrm` and so on. These are defined using the `\g_@@_mathrm_tl(...)` macros, which default to `\rmdefault` but may be specified with the `\setmathrm(...)` commands in the preamble.

Since L^AT_EX only generally defines one level of boldness, we omit `\mathbf` in the bold maths series. It can be specified as per usual with `\setboldmathrm`, which stores the appropriate family name in `\g_@@_bfmathrm_tl`.

```

74 \DeclareSymbolFont{operators}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\shapedefault
75 \SetSymbolFont{operators}{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\shapedefault
76 \DeclareSymbolFontAlphabet\mathrm{operators}
77 \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\itdefault
78 \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\shapedefault
79 \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g_@@_mathsf_tl\mddefault\shapedefault
80 \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g_@@_mathtt_tl\mddefault\shapedefault
81 \SetSymbolFont{operators}{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\shapedefault
82 \tl_if_empty:NTF \g_@@_bfmathrm_tl
83 {
84   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\itdefault
85 }
86 {
87   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mddefault\shapedefault
88   \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\bfdefault\shapedefault

```

```

89   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mdefault\itdefault
90   }
91   \SetMathAlphabet\mathsf{bold}\g_fontspec_encoding_tl\g_@@_mathsf_tl\bfdefault\shapedefault
92   \SetMathAlphabet\mathtt{bold}\g_fontspec_encoding_tl\g_@@_mathtt_tl\bfdefault\shapedefault
93   }

```

(End of definition for `\fontspec_setup_maths`:. This function is documented on page ??.)

`\fontspec_maybe_setup_maths`: We're a little less sophisticated about not executing the maths setup if various other maths font packages are loaded. This list is based on the wonderful 'L^AT_EX Font Catalogue': <http://www.tug.dk/FontCatalogue/mathfonts.html>. I'm sure there are more I've missed. Do the T_EX Gyre fonts have maths support yet?

Untested: would `\unless\ifnum\Gamma=28672\relax\bool_set_false:N \g_@@_math_bool\fi` be a better test? This needs more cooperation with euler and lucida, I think.

```

94 \cs_new:Nn \fontspec_maybe_setup_maths:
95 {
96   \@ifpackageloaded{anttor}
97   {
98     \ifx\define@antt@mathversions a\bool_gset_false:N \g_@@_math_bool\fi
99   }{}
100  \@ifpackageloaded{arevmath}      {\bool_gset_false:N \g_@@_math_bool}{}
101  \@ifpackageloaded{eulervm}      {\bool_gset_false:N \g_@@_math_bool}{}
102  \@ifpackageloaded{mathdesign}    {\bool_gset_false:N \g_@@_math_bool}{}
103  \@ifpackageloaded{concmath}     {\bool_gset_false:N \g_@@_math_bool}{}
104  \@ifpackageloaded{cmbright}     {\bool_gset_false:N \g_@@_math_bool}{}
105  \@ifpackageloaded{mathesf}      {\bool_gset_false:N \g_@@_math_bool}{}
106  \@ifpackageloaded{gfsartemisia} {\bool_gset_false:N \g_@@_math_bool}{}
107  \@ifpackageloaded{gfsneohellenic} {\bool_gset_false:N \g_@@_math_bool}{}
108  \@ifpackageloaded{iwona}
109  {
110    \ifx\define@iwona@mathversions a\bool_set_false:N \g_@@_math_bool\fi
111  }{}
112  \@ifpackageloaded{kpfonts}{\bool_gset_false:N \g_@@_math_bool}{}
113  \@ifpackageloaded{kmath}  {\bool_gset_false:N \g_@@_math_bool}{}
114  \@ifpackageloaded{kurier}
115  {
116    \ifx\define@kurier@mathversions a\bool_set_false:N \g_@@_math_bool\fi
117  }{}
118  \@ifpackageloaded{fouriernc}  {\bool_gset_false:N \g_@@_math_bool}{}
119  \@ifpackageloaded{fourier}   {\bool_gset_false:N \g_@@_math_bool}{}
120  \@ifpackageloaded{lmodern}   {\bool_gset_false:N \g_@@_math_bool}{}
121  \@ifpackageloaded{mathpazo}  {\bool_gset_false:N \g_@@_math_bool}{}
122  \@ifpackageloaded{mathptmx}  {\bool_gset_false:N \g_@@_math_bool}{}
123  \@ifpackageloaded{MinionPro} {\bool_gset_false:N \g_@@_math_bool}{}
124  \@ifpackageloaded{unicode-math} {\bool_gset_false:N \g_@@_math_bool}{}
125  \@ifpackageloaded{breqn}     {\bool_gset_false:N \g_@@_math_bool}{}
126  \@ifpackageloaded{pxfonts}   {\bool_gset_false:N \g_@@_math_bool}{}
127  \@ifpackageloaded{txfonts}   {\bool_gset_false:N \g_@@_math_bool}{}
128  \@ifpackageloaded{newpxmath} {\bool_gset_false:N \g_@@_math_bool}{}
129  \@ifpackageloaded{newtxmath} {\bool_gset_false:N \g_@@_math_bool}{}
130  \@ifpackageloaded{mtpro2}   {\bool_gset_false:N \g_@@_math_bool}{}

```

```
131 \bool_if:NT \g_@@_math_bool
132 {
133   \@@_info:n {setup-math}
134   \fontspec_setup_maths:
135 }
136 }
137 \AtBeginDocument{\fontspec_maybe_setup_maths:}
```

(End of definition for \fontspec_maybe_setup_maths:. This function is documented on page ??.)

File XX

fontspec-code-closing.dtx

1 Closing code

1.1 Finishing up

Now we just want to set up loading the .cfg file, if it exists.

```
1 \bool_if:NT \g_@@_cfg_bool
2   {
3     \InputIfFileExists{fontspec.cfg}
4     {}
5     { \typeout{No~ fontspec.cfg~ file~ found;~ no~ configuration~ loaded.} }
6   }
```

File XXI

fontspec-code-xfss.dtx

1 Changes to the NFSS

1 (*fontspec)

1.1 Italic small caps and so on

2 \providecommand*\scitdefault{\scdefault\itdefault}

3 \providecommand*\scsldefault{\scdefault\sldefault}

4 \providecommand*\scswdefault{\scdefault\swdefault}

TeX's 'shape' font axis needs to be overloaded to support italic small caps and slanted small caps. These are the combinations to support:

5 \cs_new:Nn \@@_shape_merge:nn { c_@@_shape_#1_#2_tl }

6 \cs_new:Nn \@@_merge_default_shapes:

7 {

8 \tl_const:cn { \@@_shape_merge:nn \shapedefault\scdefault } {\scdefault}

9 \tl_const:cn { \@@_shape_merge:nn \itdefault \scdefault } {\scitdefault}

10 \tl_const:cn { \@@_shape_merge:nn \sldefault \scdefault } {\scsldefault}

11 \tl_const:cn { \@@_shape_merge:nn \swdefault \scdefault } {\scswdefault}

12 \tl_const:cn { \@@_shape_merge:nn \scdefault \itdefault } {\scitdefault}

13 \tl_const:cn { \@@_shape_merge:nn \scdefault \sldefault } {\scsldefault}

14 \tl_const:cn { \@@_shape_merge:nn \scdefault \swdefault } {\scswdefault}

15 \tl_const:cn { \@@_shape_merge:nn \scsldefault \itdefault } {\scitdefault}

16 \tl_const:cn { \@@_shape_merge:nn \scitdefault \sldefault } {\scsldefault}

17 \tl_const:cn { \@@_shape_merge:nn \scitdefault \shapedefault } {\scdefault}

18 \tl_const:cn { \@@_shape_merge:nn \scsldefault \shapedefault } {\scdefault}

19 }

20 \@@_merge_default_shapes:

The following is rather specific; it only returns true if the merged shape exists, but more importantly also if the merged shape is defined for the current font.

21 \prg_new_conditional:Nnn \@@_if_merge_shape:n {TF}

22 {

23 \bool_lazy_and:nnTF

24 { \tl_if_exist_p:c { \@@_shape_merge:nn {\f@shape} {#1} } }

25 {

26 \cs_if_exist_p:c

27 {

28 \f@encoding/\f@family/\f@series/

29 \tl_use:c { \@@_shape_merge:nn {\f@shape} {#1} }

30 }

31 }

32 \prg_return_true: \prg_return_false:

33 }

34 \cs_set_eq:NN \emfontdeclare \DeclareEmphSequence

1.2 Strong emphasis

\strongfontdeclare

```

35 \cs_set_protected:Npn \strongfontdeclare #1
36 {
37   \prop_gclear:N \g_@@_strong_prop
38   \int_zero:N \l_@@_strongdef_int
39
40   \group_begin:
41     \normalfont
42     \clist_map_inline:nn {\strongreset,#1}
43     {
44       ##1
45       \prop_gput_if_new:NxV \g_@@_strong_prop { \f@series } { \l_@@_strongdef_int }
46       \prop_gput:Nxn \g_@@_strong_prop { switch-\int_use:N \l_@@_strongdef_int } { ##1 }
47       \int_incr:N \l_@@_strongdef_int
48     }
49   \group_end:
50 }

```

(End of definition for `\strongfontdeclare`. This function is documented on page ??.)

`\strongenv`

```

51 \DeclareRobustCommand \strongenv
52 {
53   \@nomath\strongenv
54
55   <debug> \typeout{Strong~ level:~\int_use:N \l_@@_strong_int}
56   \prop_get:NxNT \g_@@_strong_prop { \f@series } \l_@@_strong_tmp_tl
57   {
58     \int_set:Nn \l_@@_strong_int { \l_@@_strong_tmp_tl }
59   <debug> \typeout{Series~ (\f@series)~ detected;~ new~ level:~\int_use:N \l_@@_strong_int}
60   }
61
62   \int_incr:N \l_@@_strong_int
63
64   \prop_get:NxNTF \g_@@_strong_prop { switch-\int_use:N \l_@@_strong_int } \l_@@_strong_swit
65   { \l_@@_strong_switch_tl }
66   {
67     \int_zero:N \l_@@_strong_int
68     \strongreset
69   }
70 }
71 }

```

(End of definition for `\strongenv`. This function is documented on page ??.)

`\strong`

```

\strongreset 72 \DeclareTextFontCommand{\strong}{\strongenv}
73 \cs_set:Npn \strongreset {}

```

(End of definition for `\strong` and `\strongreset`. These functions are documented on page ??.)

`\reset@font` Ensure nesting resets when necessary:

```

74 \cs_set:Npn \reset@font

```

```
75 {
76   \normalfont
77   \int_zero:N \l_@@_strong_int
78 }
```

(End of definition for \reset@font. This function is documented on page ??.)

Programmer's interface for setting nesting levels:

```
79 \cs_new:Nn \fontspec_set_strong_level:n { \int_set:Nn \l_@@_strong_int {#1} }
```

Defaults:

```
80 \strongfontdeclare{ \bfseries }
```

```
81 </fontspec>
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
\#	261, 262, 293
\,	1, 2, 3, 4, 5, 28
\-	718
@@ commands:	
\@@_DeclareFontShape:nnnnnn	512, 519, 529, 529, 538, 547, 560, 568, 581
\g_@@_OT_features_prop	20, 22, 77
\@@_add_nfssfont:nnnn	313, 314, 315, 316, 317, 318, 319, 320, 321, 334, 334
\@@_aff_error:n	11, 397, 438, 470
\l_@@_alias_bool	25, 206, 213, 219, 224, 231, 251
\l_@@_all_features_clist	21, 57, 98, 108, 122, 190, 279
\g_@@_all_keyval_modules_clist	1, 51, 208, 226
\g_@@_all_opentype_feature_names_prop	78, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354
\l_@@_arg_clist	63, 308, 309, 310, 313, 316
\l_@@_atsui_bool	7, 10, 211, 353, 362, 655
\l_@@_basename_tl	42, 10, 95, 331
\l_@@_bf_series_seq	49, 167, 179, 182
\g_@@_bfmathrm_tl	71, 72, 82, 87, 88, 89, 126
\@@_calc_scale:n	328, 329, 341, 341
\@@_calc_scale:nn	330, 362, 362
\g_@@_cfg_bool	1, 8, 10, 19
\l_@@_check_bool	5, 52, 53, 206, 211, 217, 228
\@@_check_lang:Nn	132
\@@_check_lang:NnTF	97, 132, 337
\@@_check_lang:Nnn	136
\@@_check_lang:NnnTF	110, 132, 134
\@@_check_ot_feat:Nn	180
\@@_check_ot_feat:NnTF	50, 71, 180, 647
\@@_check_ot_feat:Nnnn	185
\@@_check_ot_feat:NnnnTF	67, 180, 182
\@@_check_script:Nn	86
\@@_check_script:NnTF	80, 86, 291
\@@_combo_sc_shape:n	520, 523, 569, 608, 616
\@@_construct_font_call:nn	135, 137, 141, 144, 150, 157, 176, 282, 400, 401, 423, 506
\@@_construct_font_call:nnnnnn	150, 159
\l_@@_curr_bfname_tl	98, 177, 187, 190, 192, 227
\l_@@_curr_fontname_tl	97, 325, 326
\g_@@_curr_series_tl	96, 166, 181, 185, 190, 192, 227, 714
\@@_declare_shape:nnnn	411, 431, 431, 449
\@@_declare_shape_loginfo:nn	447, 589, 589
\@@_declare_shape_slanted:nn	445, 539, 539
\@@_declare_shapes_bx:nn	446, 551, 551
\@@_declare_shapes_normal:nn	443, 510, 510
\@@_declare_shapes_smcaps:nn	444, 515, 515
\g_@@_default_fontopts_clist	50, 110, 121
\@@_define_aat_feature:nnnn	2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 60, 61, 62, 63, 65, 66, 67, 101, 102, 103, 104, 105, 106,

107, 109, 110, 111, 112, 113, 114, 115, 117, 118, 119, 120, 121, 122, 123, 124, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 185	\l_@@_external_bool . 26, 28, 46, 56, 384
\@@_define_aat_feature_group:n 1, 1, 1, 29, 35, 44, 55, 59, 64, 68, 80, 91, 100, 108, 116, 125, 180	\l_@@_external_kpse_bool .. 30, 30, 55
\@@_define_opentype_feature:nnnn 7, 16, 26, 29, 42, 54, 55, 56, 60, 61, 61, 77, 93, 105, 111, 112, 113, 115, 120, 121, 122, 125, 126, 127, 129, 153, 154, 157, 177, 195	\@@_extract_all_features: 93
\@@_define_opentype_feature_ group:n 6, 12, 12, 28, 41, 60, 76, 92, 104, 114, 124, 128, 156, 176, 190, 194, 203, 222, 236, 258, 268	\@@_extract_all_features:n ... 20, 93
\@@_define_opentype_onoffreset:nnnn 13, 14, 15, 16, 17, 18, 27, 34, 35, 36, 37, 38, 39, 40, 50, 51, 52, 52, 53, 54, 55, 59, 70, 71, 72, 73, 74, 75, 86, 87, 88, 89, 90, 91, 100, 101, 102, 103, 110, 123, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 155, 168, 169, 170, 171, 172, 173, 174, 175, 187, 188, 189, 190, 191, 192, 193, 195, 196, 197, 198, 199, 200, 201, 202	\l_@@_fake_ambolden_tl 134, 654, 657, 671
\@@_define_opentype_onreset:nnnn 58, 58	\l_@@_fake_slant_tl . 133, 649, 676, 679
\@@_define_opentype_variation_ axis:nn 1, 1, 577, 584, 585	\l_@@_family_fontopts_clist 56, 104, 105, 111
\g_@@_defined_shapes_tl 93, 193, 591, 713	\g_@@_family_int_prop ... 82, 256, 262
\l_@@_defining_encoding_bool . 3, 9, 15, 21, 27, 27, 50, 52, 57, 63, 69, 75	\l_@@_family_label_tl 104, 106, 132, 150, 164
\l_@@_disable_defaults_bool 24, 96, 155	\@@_feat_off:n 50, 55
\l_@@_em_int 40	\@@_feat_prop_add:nn 1, 2, 3, 4, 5, 16, 28
\g_@@_em_normalise_slant_bool ... 29	\@@_feat_reset:n 51, 56, 61
\g_@@_em_prop 79	\@@_find_autofonts: 270, 290, 290
\l_@@_em_switch_tl 118	\l_@@_firsttime_bool 1, 29, 186, 255, 273, 394, 490, 514, 527, 539, 601, 647, 666, 669, 707
\l_@@_em_tmp_tl 123	\@@_font_is_file: 35, 167, 172, 175, 182
\l_@@_emdef_int 41	\@@_font_is_kpse: 32, 58, 167, 177, 182
\l_@@_emshape_query_tl 117	\@@_font_is_name: 167, 167, 708
\@@_error:n 1, 10, 457	\l_@@_font_path_tl 29, 30, 101, 174, 709
\@@_error:nn 2, 3, 4, 10, 14, 16, 22, 28, 58, 64, 70, 76, 139, 424, 751, 770	\@@_font_suppress_not_found_ error: 5, 9, 9, 38, 269
\@@_error:nnn 4, 466	\l_@@_fontcfg_bool ... 12, 13, 18, 22, 81
\l_@@_ext_filename_tl 84, 85, 88, 89, 99	\l_@@_fontface_cs_tl 17, 71, 143, 147, 149, 154, 159, 160, 163, 167, 291, 337, 350, 422, 461, 647, 658
\l_@@_extension_tl 45, 51, 53, 72, 100, 131, 161	\l_@@_fontfeat_bf_clist 66, 225, 314, 672
\l_@@_extensions_clist 48, 54, 61, 256	\l_@@_fontfeat_bfit_clist 68, 235, 318, 656, 658, 678, 680
	\l_@@_fontfeat_bfsl_clist 70, 243, 319
	\l_@@_fontfeat_bfsw_clist 72, 251, 320
	\l_@@_fontfeat_clist 61, 129, 200, 274
	\l_@@_fontfeat_curr_clist 62, 466, 475, 488
	\l_@@_fontfeat_it_clist 67, 231, 315, 650
	\l_@@_fontfeat_sc_clist . 73, 257, 466
	\l_@@_fontfeat_sl_clist . 69, 239, 316
	\l_@@_fontfeat_sw_clist . 71, 247, 317
	\l_@@_fontfeat_up_clist 65, 221, 263, 313
	\g_@@_fontid_family_prop 81, 236, 264
	\l_@@_fontid_tl 42, 21, 23, 102, 232, 236, 244
	\l_@@_fontname_bf_tl 75, 136, 187, 295, 301, 314, 673

<p> \l_@@_fontname_bfit_tl 76, 138, 198, 294, 295, 296, 318, 659, 681 \l_@@_fontname_bfsl_tl 140, 202, 309, 319 \l_@@_fontname_bfsw_tl . 142, 206, 320 \l_@@_fontname_it_tl 74, 137, 153, 294, 306, 315, 651 \l_@@_fontname_sc_tl 143, 216, 470, 482 \l_@@_fontname_sl_tl 139, 158, 309, 316 \l_@@_fontname_sw_tl . . . 141, 162, 317 \l_@@_fontname_tl . . 103, 154, 156, 160 \l_@@_fontname_up_tl 42, 45, 9, 125, 135, 135, 137, 139, 141, 143, 144 \@@_fontname_wrap:n 47, 152, 153, 169, 170, 174, 179 \l_@@_fontopts_clist 55, 101, 102, 112, 411, 419, 420, 421 \g_@@_fontopts_prop 74, 86, 101, 104, 134, 137, 141, 142, 419 \@@_format_axis:nn 684, 698 \@@_get_features:Nn 61, 196 \@@_get_features:n . . . 28, 196, 275, 495 \@@_get_variations: 60, 283, 496, 507, 683, 688 \l_@@_graphite_bool . 11, 211, 356, 374 \l_@@_harfbuzz_bool 10, 89, 112 \c_@@_hexcol_tl 160, 224, 731 \l_@@_hexcol_tl . 152, 223, 225, 226, 476, 481, 485, 500, 505, 509, 524, 731 \l_@@_hyphenchar_tl 151, 458, 459, 461, 464 \@@_if_autofont:nn 397 \@@_if_autofont:nnTF 390 \@@_if_detect_external:n 58 \@@_if_detect_external:nTF 12, 58, 175 \@@_if_font_feature:n 265 \@@_if_font_feature:nTF 263 \@@_if_merge_shape:n 21 \@@_if_merge_shape:nTF 191 \@@_info:n 8, 133, 333, 339 \@@_info:nn 9, 392 \@@_info:nnn 10, 273 \@@_init: 6, 174, 270, 703, 703 \@@_init_fontface: 199, 724, 724 \@@_init_ttc:n 17, 70, 70 \g_@@_instance_tl 159, 617, 690, 692, 728 \@@_int_mult_truncate:Nn . 61, 61, 536 \@@_iv_str_to_num:Nn 97, 146, 147, 195, 199, 200, 754, 755, 760 \@@_iv_str_to_num:w 764, 765, 767 </p>	<p> \@@_keys_define_code:nnn . . 7, 13, 16, 20, 24, 42, 43, 52, 53, 61, 119, 124, 129, 136, 141, 145, 156, 160, 164, 169, 196, 200, 204, 208, 219, 223, 229, 233, 237, 241, 245, 249, 253, 260, 265, 271, 275, 279, 283, 287, 291, 292, 293, 294, 295, 296, 297, 301, 305, 324, 335, 354, 392, 418, 439, 443, 447, 471, 533, 550, 554, 560, 572, 579, 586, 611, 615, 687, 691 \l_@@_keys_leftover_clist 59, 123, 126, 127, 128, 201, 202, 206, 207, 210, 212, 216, 217 \@@_keys_set_known:nnN 54, 54, 60, 121, 126, 128, 200, 202, 340, 409 \l_@@_lang_name_tl 127, 132, 148, 149, 190, 192 \l_@@_lang_tl 48, 147, 182, 288, 339, 356, 632, 641 \l_@@_language_int 34, 45, 198, 199, 203, 209, 286, 340, 357 \l_@@_leftover_clist 58, 409, 411 \@@_load_external_fontoptions:N 18, 79, 79, 418 \@@_load_font: 26, 131, 131 \@@_load_fontname:Nn 410, 414, 461, 482 \@@_lua_function:nn 65, 66 \@@_lua_function:nnn 65, 67 \@@_lua_function:nnnn 65, 68, 174 \@@_lua_function:nnnnn 65, 69, 227 \@@_main_DeclareFontExtensions:n 122, 254, 254, 258 \@@_main_IfFontFeatureActiveTF:nnn 126, 259 \@@_main_addfontfeatures:n 82, 86, 146 \@@_main_aliasfontfeature:nn 106, 203 \@@_main_aliasfontfeatureoption:nnn 110, 222 \@@_main_fontspec:nn 1, 1, 3 \@@_main_liningnums:n 137, 294 \@@_main_newAATfeature:nnnn . . 94, 177 \@@_main_newfontface:NnnN 59, 63, 67, 71, 114, 114 \@@_main_newfontfamily:NnnN 43, 47, 51, 55, 101, 101, 116 \@@_main_newfontfeature:nn . . . 90, 170 \@@_main_newopentypefeature:nnn 98, 102, 187 \@@_main_oldstylenums:n 132, 287 \@@_main_setboldmathrm:nn . . . 27, 69 \@@_main_setmainfont:nn 8, 24, 39 </p>
---	--

<code>\@@_main_setmathrm:nn</code>	23, 63		
<code>\@@_main_setmathsf:nn</code>	31, 75		
<code>\@@_main_setmathtt:nn</code>	35, 81		
<code>\@@_main_setmonofont:nn</code>	18, 50		
<code>\@@_main_setsansfont:nn</code>	13, 37		
<code>\@@_make_AAT_feature:nn</code>			
.	9, 12, 12, 76, 87		
<code>\@@_make_AAT_feature_string:Nnn</code> .	26		
<code>\@@_make_AAT_feature_string:NnnTF</code>			
.	12, 17, 26, 657		
<code>\@@_make_OT_feature:nnn</code>			
.	44, 63, 82, 211, 218, 231, 242, 265, 275		
<code>\@@_make_font_shapes:Nnnnn</code>			
.	326, 406, 406		
<code>\@@_make_ot_smallcaps:TF</code> <u>644</u> , <u>645</u> , <u>653</u>			
<code>\@@_make_smallcaps:TF</code> . .	472, 644, 650		
<code>\l_@@_mapping_tl</code>			
.	22, 23, 24, 26, 155, 220, 221, 552, 556		
<code>\g_@@_math_bool</code>			
.	4, 6, 20, 98, 100, 101, 102, 103, 104, 105, 106, 107, 110, 112, 113, 116, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131		
<code>\g_@@_math_euler_bool</code>	9, 14, 34		
<code>\g_@@_math_lucida_bool</code>			
.	13, 14, 15, 15, 40		
<code>\g_@@_mathrm_tl</code>	65,		
.	66, 74, 75, 77, 78, 81, 84, 98, 125, 129		
<code>\g_@@_mathsf_tl</code>			
.	77, 78, 79, 91, 99, 127, 130		
<code>\g_@@_mathtt_tl</code>			
.	80, 83, 84, 92, 100, 128, 131		
<code>\@@_merge_default_shapes:</code>	6, 20		
<code>\l_@@_mm_bool</code>	9, 355, 366, 594, 599		
<code>\l_@@_mode_tl</code>			
.	91, 91, 101, 105, 114, 638, 717		
<code>\@@_msg_new:nn</code>			
.	14, 19, 24, 45, 85, 91, 95, 105, 109, 113, 118, 123, 128, 134, 138, 144, 148, 152, 157, 161, 178, 183, 187, 195, 199, 203, 207, 211, 216, 221		
<code>\@@_msg_new:nnn</code> <u>16</u> , <u>29</u> , <u>38</u> , <u>49</u> , <u>59</u> , <u>67</u> , <u>75</u>			
<code>\l_@@_never_check_bool</code>			
.	32, 89, 139, 187, 272		
<code>\g_@@_nfss_enc_tl</code>	4,		
.	31, 33, 44, 46, 57, 59, 107, 109, 271, 299, 512, 519, 547, 560, 568, 581, 715		
<code>\l_@@_nfss_fam_tl</code> . .	111, 234, 249, 303		
<code>\g_@@_nfss_family_tl</code>			
.	41, 108, 163, 189, 238, 249, 250, 263, 264, 271, 277, 278, 279, 280, 285, 286, 287, 288, 512, 519, 547, 548, 560, 562, 568, 570, 581, 583		
<code>\l_@@_nfss_prop</code>	75, 190, 226		
<code>\l_@@_nfss_sc_tl</code>			
.	109, 435, 441, 487, 517, 520, 566, 618		
<code>\l_@@_nfss_tl</code> <u>110</u> , <u>434</u> , <u>440</u> , <u>462</u> , <u>513</u> , <u>606</u>			
<code>\l_@@_nfssfont_prop</code>	76, 321, 344		
<code>\l_@@_nobf_bool</code>			
.	2, 26, 173, 176, 292, 299, 674		
<code>\l_@@_noit_bool</code>			
.	3, 27, 149, 152, 292, 304, 652		
<code>\l_@@_nosc_bool</code> <u>4</u> , <u>212</u> , <u>215</u> , <u>468</u> , <u>479</u> , <u>485</u>			
<code>\c_@@_opacity_tl</code>			
.	161, 162, 224, 525, 537, 730		
<code>\l_@@_opacity_tl</code>	153,		
.	223, 225, 226, 525, 530, 537, 542, 730		
<code>\l_@@_optical_size_tl</code>			
.	154, 164, 591, 608, 710		
<code>\l_@@_options_tl</code> . . .	104, 153, 156, 160		
<code>\l_@@_ot_bool</code>	8, 28, 39,		
.	65, 78, 91, 108, 121, 136, 204, 271, 354, 370, 379, 589, 599, 629, 652, 706		
<code>\@@_ot_compat:nn</code> . . .	360, 364, 365,		
.	366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380		
<code>\@@_ot_validate_tag:n</code>	113,		
.	169, 170, 221, 222, 223, 736, 737, 741		
<code>\@@_ot_validate_tag:w</code>	739, 742		
<code>\@@_ot_validate_tag_aux:w</code> <u>745</u> , <u>746</u> , <u>748</u>			
<code>\g_@@_pkg_euler_loaded_bool</code> <u>2</u> , <u>3</u> , <u>8</u> , <u>16</u>			
<code>\c_@@_postadjust_tl</code>	163, 732		
<code>\l_@@_postadjust_tl</code>			
.	158, 441, 451, 463, 513, 520, 548, 563, 571, 584, 619, 622, 732		
<code>\l_@@_pre_feat_sclist</code>	137,		
.	144, 165, 283, 400, 401, 423, 507, 626		
<code>\@@_preparse_features:</code> . .	<u>25</u> , <u>117</u> , <u>117</u>		
<code>\l_@@_prev_unicode_name_tl</code> <u>48</u> , <u>53</u> , <u>106</u>			
<code>\l_@@_primitive_font</code>	39, 40		
<code>\@@_primitive_font_current_name:</code>			
.	<u>56</u> , <u>57</u> , <u>184</u> , <u>186</u>		
<code>\@@_primitive_font_get_name:N</code> . .			
.	<u>56</u> , <u>56</u> , <u>59</u>		
<code>\@@_primitive_font_glyph_if_-</code>			
exist:Nn	44		
<code>\@@_primitive_font_glyph_if_-</code>			
exist:NnTF	<u>44</u> , <u>461</u>		
<code>\@@_primitive_font_gset:Nnn</code>			
.	<u>1</u> , <u>5</u> , <u>26</u> , <u>28</u> , <u>33</u>		
<code>\@@_primitive_font_gset:NnnTF</code> . . .	34		
<code>\@@_primitive_font_gset:NnnTTF</code> .	<u>21</u>		

\@@_primitive_font_gset:Onn . . . 33, 143
\@@_primitive_font_gset:OnnTF . . . 34
\@@_primitive_font_gset_p:NnnTF . . . 21
\@@_primitive_font_if_exist:n . . . 35
\@@_primitive_font_if_exist:nTF
. 35, 176
\@@_primitive_font_if_null:N . . . 13
\@@_primitive_font_if_null:NTF .
. 13, 24, 29, 40
\@@_primitive_font_if_null_p:N . . . 13
\@@_primitive_font_set:Nnn
. 1, 1, 21, 23, 31, 39, 400, 401
\@@_primitive_font_set:NnnTF . 32, 136
\@@_primitive_font_set:NnnTTF . . 21
\@@_primitive_font_set:Onn 31
\@@_primitive_font_set:OnnTF . 32, 422
\@@_primitive_font_set_hyphenchar:Nn
. 52, 52, 452, 464
\@@_primitive_font_set_p:NnnTF . . 21
\l_@@_proceed_bool 31, 67, 74, 80
\l_@@_punctspace_adjust_tl
. 156, 163, 424, 429, 434, 734
\g_@@_rawfeatures_sclist 59,
164, 278, 283, 496, 507, 670, 679, 726
\g_@@_rawvariations_prop
. 6, 83, 613, 694, 698, 727
\@@_remove_clashing_featstr:n . .
. 34, 78, 673, 673, 682
\l_@@_renderer_tl 52,
70, 74, 75, 90, 163, 363, 371, 375, 712
\l_@@_rmfamily_encoding_tl
. 9, 15, 21, 31, 169
\l_@@_rmfamily_family_tl . . 29, 30, 166
\@@_sanitise_fontname:Nn
. 8, 9, 10, 44, 44, 74, 75, 76, 130, 417
\@@_save_family:nn 33, 267, 267
\@@_save_family_needed:n 228
\@@_save_family_needed:nTF . . . 31, 228
\@@_save_fontid_family:nn 244, 254, 266
\@@_save_fontinfo:n 269, 275, 275
\l_@@_saved_fontname_tl 105, 436, 453
\l_@@_scale_tl
. 150, 201, 332, 337, 338, 352, 360,
366, 368, 370, 374, 498, 500, 505, 729
\l_@@_script_int 33, 42,
94, 147, 149, 155, 200, 203, 209, 285, 296
\l_@@_script_name_tl 122, 132,
142, 145, 146, 188, 190, 191, 294, 308
\l_@@_script_tl
. 47, 95, 134, 144, 182, 287, 295, 631, 640
\l_@@_scriptlang_exist_bool
. 28, 288, 297, 303, 334, 342, 346
\@@_select_font_family:nn
. 1, 1, 43, 151, 156, 159, 165
\@@_set_autofont:Nnn
. 294, 295, 296, 301, 306, 309, 382, 382
\@@_set_default_features:nn
. 76, 118, 118
\@@_set_faces: 272, 311, 311
\@@_set_faces_aux:nnnnn 321, 323
\@@_set_family:NnnN 147, 156, 157
\@@_set_font_default_features:nnn
. 77, 123, 123
\@@_set_font_dimen:NnN
. 349, 350, 376, 376
\@@_set_font_type:N 9, 27, 38,
64, 77, 90, 107, 120, 135, 142, 348, 348
\@@_set_fontface:NNnnN . . 161, 169, 170
\@@_set_scriptlang: 27, 183, 183
\@@_setboldmathrm_hook:nn . . . 73, 93
\@@_setmainfont_hook:nn 34, 87
\@@_setmathrm_hook:nn 67, 90
\@@_setmathsf_hook:nn 79, 91
\@@_setmathtt_hook:nn 85, 92
\@@_setmonofont_hook:nn 60, 89
\@@_setsansfont_hook:nn 47, 88
\@@_setup_nfss:Nnnn . 462, 487, 491, 491
\@@_setup_single_size:nn 438, 450, 450
\l_@@_sffamily_encoding_tl
. 10, 17, 22, 44, 170
\l_@@_sffamily_family_tl . . 42, 43, 167
\@@_shape_merge:nn
. 5, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 24, 29, 193, 525, 526
\l_@@_shaper_tl 92, 96, 101, 106, 115, 639
\g_@@_single_feat_tl . . 75, 94, 103,
267, 279, 281, 283, 298, 341, 358, 668
\l_@@_size_tl
. 112, 285, 452, 457, 458, 493, 505
\l_@@_sizedfont_tl
. 113, 289, 453, 461, 463
\l_@@_sizefeat_clist
. 52, 53, 262, 267, 339, 345
\l_@@_sizing_leftover_clist
. 60, 456, 462, 488
\l_@@_smcp_shape_tl
. 116, 193, 196, 199, 202
\@@_strip_leading_sign:Nw . . 758, 761
\@@_strip_plus_minus:n 196, 198
\@@_strip_plus_minus_aux:Nq . 198, 199

89, 104, 112, 149, 155, 159, 173, 211,
212, 213, 231, 271, 272, 297, 342,
359, 362, 366, 370, 374, 379, 479, 707
\bool_until_do:nn 101, 152, 207
\breve 23

C

char commands:
\char_set_catcode_ignore:n 229
\char_set_catcode_space:n 18
\check 24
clist commands:
\clist_clear:N 102, 105, 274, 420
\clist_count:N 310
\clist_count:n 101
\clist_gput_right:Nn 120
\clist_gset:Nn 1, 120
\clist_map_break: 54, 66, 299, 343
\clist_map_inline:Nn 48, 61, 208, 226
\clist_map_inline:nn 42, 74,
86, 126, 210, 229, 250, 289, 335, 438, 676
\clist_new:N 50, 51,
52, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73
\clist_put_left:Nn 475
\clist_put_right:Nn 221,
225, 231, 235, 239, 243, 247, 251,
257, 263, 650, 656, 658, 672, 678, 680
\clist_set:Nn
. 53, 98, 108, 256, 262, 267, 308, 339
\clist_set_eq:NN 309, 466
\colon 120, 30, 31

color commands:
\color_export:nnN 476
\color_if_exist:nTF 474, 498
\convertcolorspec 481, 505
cs commands:

\cs:w 129
\cs_end: 129
\cs_generate_variant:Nn
. 12, 13, 60, 71, 72, 73, 74, 75, 76,
77, 78, 79, 80, 81, 82, 82, 83, 84, 85,
86, 158, 266, 449, 538, 682, 683, 741, 760
\cs_if_eq:NNTF 114, 171, 224, 402
\cs_if_exist:NTF 3, 7, 32, 199, 479, 503
\cs_if_exist_p:N 26
\cs_new:Nn 1, 1, 1, 4, 5, 5, 6, 7, 11,
12, 12, 14, 16, 16, 24, 26, 37, 44, 50,
50, 51, 52, 52, 54, 58, 61, 63, 63, 69,
70, 75, 79, 79, 81, 83, 93, 94, 101, 114,
117, 118, 123, 131, 146, 146, 147,
150, 156, 157, 157, 161, 167, 169,

170, 170, 172, 177, 177, 183, 187,
196, 198, 203, 222, 254, 254, 259,
267, 275, 283, 290, 311, 323, 328,
330, 334, 341, 348, 360, 362, 376,
382, 406, 414, 431, 450, 491, 510,
515, 523, 529, 539, 551, 589, 644,
645, 650, 663, 673, 684, 688, 724, 755
\cs_new:Npn 1, 2, 3, 4, 5,
6, 7, 8, 9, 10, 11, 52, 53, 57, 84, 199, 239
\cs_new_eq:NN 56, 182
\cs_new_protected:Nn 1, 287, 294, 737
\cs_set:Npn 1, 5, 9, 31, 32,
33, 34, 66, 67, 68, 69, 73, 74, 170, 174,
179, 398, 522, 703, 742, 748, 761, 767
\cs_set_eq:NN 34,
43, 87, 88, 89, 90, 91, 92, 93, 169, 180
\cs_set_protected:Npn 35
\cs_to_str:N 103, 108, 116, 129, 173, 226
\cs_undefine:N 77, 250, 534
\cyrillicencoding 45, 49, 88

D

\date 55
\ddot 20
\DeclareDocumentCommand 1,
7, 13, 19, 25, 31, 51, 55, 61, 67, 67, 73
\DeclareEmphSequence 34
\DeclareFontEncoding 33, 35
\DeclareFontExtensions 120
\DeclareFontFamily 35, 271
\DeclareFontSeriesDefault
. 26, 27, 39, 40, 52, 53
\DeclareFontShape
. 53, 54, 37, 39, 41, 43, 45, 536
\DeclareFontSubstitution 34, 36
\DeclareKeys 1
\DeclareMathAccent
. 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
\DeclareMathDelimiter 65, 66, 67, 68, 69
\DeclareMathSymbol
. 31, 36, 37, 38, 39, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54,
55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 70
\DeclareRobustCommand 51
\DeclareSymbolFont 16, 74
\DeclareSymbolFontAlphabet 76
\DeclareTextCommand 5, 11
\DeclareTextComposite 23
\DeclareTextCompositeCommand 29
\DeclareTextFontCommand 72
\DeclareTextSymbol 17
\DeclareUnicodeEncoding 22, 31

<code>\def</code>	27	file commands:	
<code>\defaultfontfeatures</code>	<u>73</u>	<code>\file_if_exist:nTF</code>	88
<code>\Delta</code>	53	<code>\file_input:n</code>	89
dim commands:		<code>\filedate</code>	55
<code>\dim_compare:nNnTF</code>	379	<code>\fileversion</code>	55
<code>\dim_eval:n</code>	3, 7	<code>\fmtname</code>	28
<code>\dim_new:N</code>	46, 47, 48	<code>\font</code> . 42, 3, 7, 9, 12, 27, 38, 50, 64, 67, 77,	
<code>\dim_set:Nn</code>	378	80, 90, 97, 107, 110, 114, 120, 135,	
<code>\dim_to_fp:n</code>	354, 355	171, 224, 349, 383, 404, 405, 406,	
dim internal commands:		412, 413, 414, 425, 430, 435, 452, 464	
<code>__dim_eval:w</code>	63	<code>\fontdimen</code>	84, 378, 404,
<code>__dim_eval_end:</code>	63	405, 406, 412, 413, 414, 425, 430, 435	
<code>\dot</code>	26	<code>\fontdimen8</code>	83
<code>\DTX</code>	3, 4, 5, 6, 7, 8, 9, 10, 11, 12,	<code>\fontencoding</code> 4, 9, 10, 11, 15, 17, 19, 109, 345	
	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23	<code>\fontfamily</code>	
		30, 14, 15, 16, 17, 18, 19, 108, 163, 346	
		<code>\fontname</code>	40, 56
		<code>\fontspec</code>	24, 31, 42, 1
		fontspec commands:	
		<code>\fontspec_calc_scale:n</code>	82, 83
		<code>\fontspec_calc_scale:nn</code>	82
		<code>\fontspec_complete_fontname:Nn</code> .	
	 143, 153, 158, 162, 177,	
		198, 202, 206, 216, 289, 325, <u>328</u> , 328	
		<code>\g_fontspec_default_fontopts_tl</code> .	31
		<code>\g_fontspec_encoding_tl</code>	38,
		42, 43, 45, 46, 49, 50, 74, 75, 77, 78,	
		79, 80, 81, 84, 85, 87, 88, 89, 91, 92, 715	
		<code>\l_fontspec_family_tl</code>	
	 42, 41, 84, 153, 166	
		<code>\l_fontspec_feature_string_tl</code> .	19, 53
		<code>\l_fontspec_font</code> 42, 149, 154, 160, 167	
		<code>\fontspec_font_if_exist:n</code> ..	<u>171</u> , 171
		<code>\fontspec_font_if_exist:nTF</code>	180
		<code>\l_fontspec_fontname_tl</code> 42, 45, 8, 18,	
		21, 86, 101, 116, 121, 125, 126, 131,	
		141, 201, 282, 296, 301, 306, 309,	
		313, 410, 428, 433, 436, 461, 466,	
		482, 493, 506, 602, 651, 659, 673, 681	
		<code>\fontspec_gset_family:Nnn</code>	
	 65, 66, 71, 72, 77, 78, 83, 84, 156	
		<code>\fontspec_gset_fontface:NNnn</code> ...	169
		<code>\fontspec_if_aat_feature:nn</code>	5
		<code>\fontspec_if_aat_feature:nnTF</code>	<u>5</u>
		<code>\fontspec_if_current_feature:n</code> .	181
		<code>\fontspec_if_current_feature:nTF</code>	
	 <u>181</u> , 283	
		<code>\fontspec_if_current_language:n</code> .	131
		<code>\fontspec_if_current_language:nTF</code>	
	 <u>131</u>	
		<code>\fontspec_if_current_script:n</code> ..	116
E			
<code>\else</code>	17, 106, 213, 776, 777		
else commands:			
<code>\else:</code>	48		
<code>\emfontdeclare</code>	34		
<code>\EncodingAccent</code>	<u>7</u>		
<code>\EncodingCommand</code>	<u>1</u>		
<code>\EncodingComposite</code>	19		
<code>\EncodingCompositeCommand</code>	<u>25</u>		
<code>\encodingdefault</code> 82, 21, 22, 23, 33, 46, 59, 345			
<code>\EncodingSymbol</code>	<u>13</u>		
<code>\endinput</code>	8, 13, 22, 28		
exp commands:			
<code>\exp_after:wN</code>	455		
<code>\exp_args:NNNx</code>	358, 372		
<code>\exp_args:Nnnx</code>	195		
<code>\exp_args:Nnx</code>	54, 55, 56, 60, 61		
<code>\exp_args:No</code>			
..... 17, 71, 88, 291, 337, 461, 647, 657			
<code>\exp_args:NV</code>	283		
<code>\exp_args:Nx</code>	64, 438		
<code>\exp_args:Nxx</code>	185		
<code>\exp_last_unbraced:No</code> ..	31, 32, 33, 34		
<code>\exp_not:N</code>	21, 83, 106, 108,		
109, 110, 239, 240, 243, 244, 252,			
253, 278, 279, 416, 594, 608, 621, 622			
<code>\exp_not:n</code> ...	60, 65, 198, 262, 593, 607		
<code>\expandafter</code>	28		
F			
<code>\familydefault</code>	32, 45, 58, 346		
<code>\fi</code>	19, 28, 28, 32, 41, 54,		
98, 108, 110, 116, 215, 367, 376, 776, 777			
fi commands:			
<code>\fi:</code>	50		

<code>\fontspec_if_current_script:nTF</code>	116
<code>\fontspec_if_feature:n</code>	34
<code>\fontspec_if_feature:nnn</code>	60
<code>\fontspec_if_feature:nnnTF</code>	60
<code>\fontspec_if_feature:nTF</code>	34
<code>\fontspec_if_fontspec_font:</code>	1
<code>\fontspec_if_fontspec_font:TF</code>	1 , 7 , 25 , 36 , 62 , 75 , 88 , 105 , 118 , 133 , 149
<code>\fontspec_if_language:n</code>	86
<code>\fontspec_if_language:nn</code>	103
<code>\fontspec_if_language:nnTF</code>	103
<code>\fontspec_if_language:nTF</code>	86
<code>\fontspec_if_opentype:</code>	23
<code>\fontspec_if_opentype:TF</code>	23
<code>\fontspec_if_script:n</code>	73
<code>\fontspec_if_script:nTF</code>	73
<code>\fontspec_if_small_caps:</code>	189
<code>\fontspec_if_small_caps:TF</code>	189
<code>\fontspec_maybe_setup_maths:</code>	94 , 94 , 137
<code>\fontspec_new_lang:nn</code>	118 , 330
<code>\fontspec_new_script:nn</code>	114 , 283
<code>\fontspec_parse_colour:niii</code>	488 , 512 , 522
<code>\fontspec_parse_cv:w</code>	239 , 252
<code>_fontspec_parse_wordspace:w</code>	395 , 398 , 398
<code>\fontspec_select:nn</code>	43 , 43
<code>\fontspec_set_family:Nnn</code>	3 , 29 , 42 , 55 , 103 , 146 , 157 , 158
<code>\fontspec_set_fontface:NNnn</code>	159 , 170
<code>\fontspec_set_strong_level:n</code>	79
<code>\fontspec_setup_maths:</code>	1 , 4 , 134
fontspec internal commands:	
<code>_fontspec_calc_scale:n</code>	365 , 367
<code>\l_fontspec_check_bool</code>	100 , 104 , 110 , 119 , 151 , 159 , 166 , 175
<code>_fontspec_update_featstr:n</code>	97
<code>\FONTSPECCTX</code>	2
<code>\FontspecSetCheckBoolFalse</code>	52
<code>\FontspecSetCheckBoolTrue</code>	52
fp commands:	
<code>\fp_eval:n</code>	338 , 354 , 370
<code>\fp_new:N</code>	44 , 45
G	
<code>\g</code>	121
<code>\Gamma</code>	52
<code>\gdef</code>	2
<code>\GetFileInfo</code>	54
<code>\global</code>	7
<code>\grave</code>	19
group commands:	
<code>\group_begin:</code>	4 , 28 , 37 , 40 , 151 , 173 , 268 , 289 , 296 , 343 , 364 , 399 , 408 , 532
<code>\group_end:</code>	33 , 39 , 41 , 42 , 49 , 162 , 177 , 178 , 276 , 292 , 299 , 359 , 373 , 403 , 404 , 412 , 535
H	
<code>\hat</code>	119 , 25
<code>\hbar</code>	71
I	
<code>\IfBooleanTF</code>	120 , 132
<code>\ifcase</code>	357
<code>\ifdefined</code>	26 , 39 , 52
<code>\IfFontExistsTF</code>	180
<code>\IfFontFeatureActiveTF</code>	124 , 259
<code>\IfNoValueTF</code>	75
<code>\ifnum</code>	103 , 209 , 364
<code>\ifx</code>	15 , 28 , 30 , 98 , 110 , 116 , 776 , 777
<code>\ignorespaces</code>	4 , 9 , 14 , 19 , 78 , 168
<code>\InputIfFileExists</code>	3
int commands:	
<code>\int_case:nn</code>	72 , 93 , 98
<code>\int_case:nnTF</code>	384
<code>\int_compare:nNnTF</code>	157
<code>\int_compare:nTF</code>	32 , 68 , 89 , 101 , 310 , 484 , 487 , 508 , 511 , 545 , 750 , 769
<code>\int_compare_p:nNn</code>	101 , 152 , 207
<code>\int_eval:n</code>	259
<code>\int_if_even:nTF</code>	37
<code>\int_incr:N</code>	47 , 62 , 107 , 163 , 214
<code>\int_new:N</code>	33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43
<code>\int_set:Nn</code>	11 , 42 , 45 , 58 , 63 , 71 , 79 , 94 , 98 , 105 , 148 , 154 , 160 , 202 , 212 , 296 , 340 , 535 , 718 , 772
<code>\int_to_hex:n</code>	546
<code>\int_use:N</code>	46 , 55 , 59 , 64
<code>\int_zero:N</code>	38 , 67 , 77 , 99 , 150 , 198 , 205 , 357 , 719 , 720 , 721
<code>\l_tmpa_int</code>	99 , 101 , 103 , 105 , 107 , 205 , 207 , 210 , 212 , 214
<code>\l_tmpb_int</code>	98 , 101 , 105 , 202 , 207 , 212
<code>\itdefault</code>	2 , 9 , 12 , 15 , 77 , 84 , 89 , 315 , 318 , 543 , 544 , 548 , 577 , 578 , 583 , 598 , 601
K	
keys commands:	
<code>\l_keys_choice_int</code>	68 , 72 , 89 , 93 , 98

201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383	
<code>\newfontscript</code> . 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170	
<code>\newICUfeature</code> 100	
<code>\newopentypefeature</code> 96, 187	
<code>\normalfont</code> 35, 41, 48, 61, 76	
<code>\normalsize</code> 44, 533	
<code>\nullfont</code> 15	
	<code>\numexpr</code> 44
	O
	<code>\oldstylenums</code> 35, 128, 287
	<code>\Omega</code> 62
	<code>\or</code> 360, 368, 372
	P
	<code>Path</code> 24
	<code>\Phi</code> 60
	<code>\Pi</code> 57
	<code>\postexhyphenchar</code> 721
	<code>\posthyphenchar</code> 719
	<code>\preexhyphenchar</code> 720
	<code>\prehyphenchar</code> 718
	prg commands:
	<code>\prg_new_conditional:Nnn</code> . 1, 5, 13, 21, 21, 23, 26, 26, 34, 35, 44, 58, 60, 73, 86, 86, 103, 116, 131, 132, 136, 171, 180, 181, 185, 189, 228, 265, 397
	<code>\prg_return_false:</code> 3, 13, 16, 18, 20, 24, 28, 29, 30, 31, 32, 41, 49, 50, 51, 53, 57, 67, 68, 69, 71, 80, 82, 84, 93, 97, 99, 101, 110, 110, 112, 114, 125, 126, 127, 129, 134, 140, 142, 143, 144, 166, 175, 178, 183, 187, 191, 204, 207, 217, 228, 239, 281, 284, 403
	<code>\prg_return_true:</code> 3, 13, 16, 24, 28, 29, 32, 42, 47, 50, 54, 67, 68, 80, 90, 97, 110, 110, 122, 125, 134, 140, 140, 166, 175, 177, 183, 187, 188, 205, 217, 228, 245, 251, 284, 404
	<code>\ProcessKeyOptions</code> 31
	prop commands:
	<code>\prop_gc clear:N</code> 37, 727
	<code>\prop_get:NnN</code> 41, 44, 47, 48, 93, 95, 123, 138, 153, 154
	<code>\prop_get:NnNTF</code> 56, 64, 78, 79, 101, 104, 134, 236, 256, 419
	<code>\prop_gput:Nnn</code> 6, 22, 46, 77, 137, 142, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 262, 263, 264, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 278, 279, 279, 280, 280, 281, 282, 283, 284, 285, 285, 286, 286, 287, 287, 288, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304,

T

T_EX and L^AT_EX 2_ε commands:

<code>\@</code>	31, 62
<code>\@filelist</code>	44
<code>\@ifpackageloaded</code>	
..	1, 6, 13, 14, 15, 96, 100, 101, 102, 103, 104, 105, 106, 107, 108, 112, 113, 114, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130
<code>\@nomath</code>	53
<code>\@onlypreamble</code>	94, 95, 96, 97
<code>\@rmfamilyhook</code>	7, 9
<code>\@sffamilyhook</code>	10
<code>\@tempa</code>	29, 30
<code>\@ttfamilyhook</code>	11
<code>\add@unicode@accent</code>	11
<code>\color@</code>	479, 503
<code>\curr@fontshape</code>	115, 172, 225
<code>\define@ant@mathversions</code>	98
<code>\define@iwona@mathversions</code>	110
<code>\define@kurier@mathversions</code>	116
<code>\f@encoding</code>	28, 199, 202, 203
<code>\f@family</code> ...	3, 3, 28, 41, 44, 47, 48, 93, 95, 123, 138, 153, 154, 199, 202, 203
<code>\f@series</code> ..	28, 45, 56, 59, 199, 202, 203
<code>\f@shape</code>	24, 29, 193
<code>\f@size</code>	39, 115, 138, 145, 172, 225, 400, 401, 423, 534
<code>\reset@font</code>	74
<code>\two@digits</code>	231, 243, 244
tex commands:	
<code>\tex_font:D</code>	59
<code>\tex_hyphenchar:D</code>	54
<code>\tex_iffontchar:D</code>	46
<code>\textsc</code>	36
<code>\textsf</code>	32
<code>\Theta</code>	54
<code>\tilde</code>	21
<code>\title</code>	31
tl commands:	
<code>\c_empty_tl</code>	
.....	62, 745, 746, 764, 765, 776, 777
<code>\tl_build_begin:N</code>	434, 435
<code>\tl_build_end:N</code>	440, 441
<code>\tl_build_put_right:Nn</code>	503
<code>\tl_clear:N</code>	
.	23, 24, 51, 106, 135, 307, 308, 317, 452, 709, 710, 711, 712, 729, 733, 734
<code>\tl_clear_new:N</code>	87, 88, 89
<code>\tl_const:Nn</code>	8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 160, 161, 162, 163

<code>\tl_count:n</code> ..	484, 487, 508, 511, 750, 769
<code>\tl_gclear:N</code> ...	267, 713, 714, 726, 728
<code>\tl_gput_right:Nn</code>	591, 670
<code>\tl_gremove_all:Nn</code>	679
<code>\tl_gset:Nn</code>	38, 75, 98, 99, 100, 103, 129, 130, 131, 166, 181, 263, 298, 299, 341, 358, 617, 668
<code>\tl_gset_eq:NN</code> ..	156, 169, 238, 249, 715
<code>\tl_if_empty:NTF</code>	
.....	29, 46, 50, 82, 106, 188, 220, 234, 268, 281, 337, 342, 363, 371, 375, 388, 457, 470, 498, 517, 566, 631, 632, 639, 640, 641, 654, 676, 690
<code>\tl_if_empty:nTF</code> I4, I8, 29, 69, 82, 83, 84, 140, 147, 171, 210, 338, 386, 400, 619	
<code>\tl_if_empty_p:N</code>	30
<code>\tl_if_empty_p:n</code>	81, 92, 142, 190
<code>\tl_if_eq:NNTF</code>	201, 525, 537
<code>\tl_if_eq:nnTF</code>	85, 185
<code>\tl_if_exist:NTF</code>	146, 525
<code>\tl_if_exist_p:N</code>	24
<code>\tl_if_in:NnTF</code>	44, 50, 313
<code>\tl_if_in:nnTF</code>	65, 185
<code>\tl_if_single:nTF</code>	128, 457
<code>\tl_new:N</code>	84, 85, 86, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 159, 164, 165, 166, 167, 168, 169, 170, 171
<code>\tl_put_left:Nn</code>	46
<code>\tl_put_right:Nn</code>	
.....	9, 10, 11, 136, 441, 451, 463
<code>\tl_remove_all:Nn</code> ...	47, 85, 243, 332
<code>\tl_remove_once:Nn</code>	52
<code>\tl_replace_all:Nnn</code> ..	14, 16, 18, 86, 331
<code>\tl_set:Nn</code>	21, 21, 22, 22, 23, 26, 28, 29, 34, 39, 39, 40, 41, 42, 45, 46, 49, 53, 53, 70, 84, 91, 96, 114, 115, 115, 116, 122, 127, 129, 133, 134, 138, 139, 146, 149, 149, 150, 154, 160, 163, 164, 167, 172, 173, 196, 225, 226, 242, 258, 261, 269, 273, 285, 294, 295, 303, 330, 332, 336, 337, 338, 339, 352, 356, 360, 363, 368, 371, 374, 375, 391,

402, 410, 424, 429, 434, 458, 459, 485, 500, 500, 509, 524, 530, 542, 552, 556, 591, 608, 626, 649, 671, 717	
<code>\tl_set_eq:NN</code>	
..... 30, 31, 33, 41, 43, 43, 44, 45, 46, 46, 48, 49, 50, 53, 56, 57, 59, 125, 157, 170, 187, 193, 316, 366, 436, 453, 651, 659, 673, 681, 730, 731, 732	
<code>\tl_tail:n</code>	683, 697
<code>\tl_to_str:N</code>	21
<code>\tl_to_str:n</code>	78, 186
<code>\tl_trim_spaces:n</code>	15, 17
<code>\tl_use:N</code>	29, 526
<code>\tmpa</code>	27, 28
token commands:	
<code>\token_to_str:N</code>	78, 479, 503
<code>\ttdefault</code>	41, 56, 58, 100, 131
<code>\ttfamily</code>	7
<code>\typeout</code> 3, 5, 23, 37, 43, 55, 56, 58, 59, 60, 65, 74, 83, 88, 95, 96, 101, 117, 119, 121, 125, 125, 133, 135, 138, 141, 148, 156, 183, 184, 185, 190, 192, 194, 198, 205, 206, 210, 212, 216, 220, 230, 231, 232, 238, 245, 261, 262, 278, 279, 287, 293, 305, 312, 315, 350, 358, 361, 365, 369, 373, 416, 433, 458, 463, 474, 478, 493, 496, 531, 665, 669, 675, 678, 705, 757	
U	
<code>\UndeclareAccent</code>	55
<code>\UndeclareCommand</code>	55
<code>\UndeclareComposite</code>	73
<code>\UndeclareSymbol</code>	55
<code>\UndeclareTextCommand</code>	59, 65, 71
<code>\unexpanded</code>	95, 125
<code>\UnicodeEncodingName</code>	5, 11, 17, 23, 29, 48, 49, 53, 59, 65, 71, 78
<code>\UnicodeFontFile</code>	38, 40, 42, 44, 46
<code>\UnicodeFontTeXLigatures</code>	38, 40, 42, 44, 46
<code>\Upsilon</code>	59
<code>\url</code>	39
use commands:	
<code>\use:N</code>	100, 108
<code>\use:n</code>	104, 157, 169, 237, 455
<code>\use_i:nnn</code>	322
<code>\use_ii:nnn</code>	322
<code>\use_iii:nnn</code>	308
<code>\use_none:nm</code> ..	87, 88, 89, 90, 91, 92, 93
<code>\UTFencname</code>	43, 87
X	
<code>\XeLaTeX</code>	33
<code>\XeTeXcountvariations</code>	364
<code>\XeTeXfeaturename</code>	28
<code>\XeTeXfonttype</code>	357
<code>\XeTeXisexclusivefeature</code>	32
<code>\XeTeXOTcountfeatures</code>	203
<code>\XeTeXOTcountlanguages</code>	149
<code>\XeTeXOTcountscripts</code>	98
<code>\XeTeXOTfeaturetag</code>	209
<code>\XeTeXOTlanguagetag</code>	155
<code>\XeTeXOTscripttag</code>	103
<code>\XeTeXselectorname</code>	34, 39, 44
<code>\Xi</code>	56