

The fontspec package

Font selection for X_YL^AT_EX and LuaL^AT_EX

WILL ROBERTSON

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

<http://latex3.github.io/fontspec/>

2024/04/27 v2.9b

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	17
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
2	Overview	36
2.1	Commands	36
2.2	Conditionals	36
3	Implementation	38
X	fontspec-code-internal.dtx	44
1	Internals	44
1.1	The main function for setting fonts	44
1.2	Setting font shapes in a family	52
1.3	Initialisation	63
1.4	Miscellaneous	63
XI	fontspec-code-opentype.dtx	66
1	OpenType definitions code	66
1.1	Adding features when loading fonts	67
1.2	OpenType feature information	71

XII	fontspec-code-graphite.dtx	75
1	Graphite/AAT code	75
XIII	fontspec-code-keyval.dtx	77
1	Font loading (keyval) definitions	77
1.1	Pre-pre-parsing stages	77
1.2	Pre-parsed features	79
1.3	Font faces	80
1.4	General font-independent features	84
XIV	fontspec-code-feat-opentype.dtx	94
1	OpenType feature definitions	94
2	Regular key=val / tag definitions	94
2.1	Ligatures	94
2.2	Letters	94
2.3	Numbers	95
2.4	Vertical position	95
2.5	Contextuals	96
2.6	Diacritics	96
2.7	Kerning	96
2.8	Fractions	97
2.9	Style	97
2.10	CJK shape	98
2.11	Character width	98
2.12	Vertical	98
3	OpenType features that need numbering	99
3.1	Alternate	99
3.2	Variant / StylisticSet	99
3.3	CharacterVariant	99
3.4	Annotation	100
3.5	Ornament	100
4	Script and Language	100
4.1	Script	100
4.2	Language	102
5	Backwards compatibility	103
XV	fontspec-code-scripts.dtx	104
1	Font script definitions	104

XVI	fontspec-code-lang.dtx	108
1	Font language definitions	108
XVII	fontspec-code-feat-aat.dtx	116
1	AAT feature definitions	116
1.1	Ligatures	116
1.2	Letters	116
1.3	Numbers	117
1.4	Contextuals	117
1.5	Diacritics	117
1.6	Vertical position	117
1.7	Fractions	117
1.8	Alternate	117
1.9	Variant / StylisticSet	118
1.10	Style	118
1.11	CJK shape	118
1.12	Character width	119
1.13	Annotation	119
XVIII	fontspec-code-enc.dtx	120
1	Extended font encodings	120
XIX	fontspec-code-math.dtx	123
1	Selecting maths fonts	123
XX	fontspec-code-closing.dtx	128
1	Closing code	128
1.1	Finishing up	128
XXI	fontspec-code-xfss.dtx	129
1	Changes/additions to the NFSS	129
2	Implementation	129
2.1	Italic small caps and so on	129
2.2	Strong emphasis	130
2.3	Defaults	131
	Index	132

File I

fontspec.dtx

1 Package declaration

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

```
1 <*dtx>
2 \gdef\FONTSPECCTX{
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 T_EX; 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://latex3.github.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 {2024/04/27}{2.9b}{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        = "2.9b",
62 <lua>  date           = "2024/04/27",
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 \LaTeX

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

X \LaTeX

```
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 {0000000}
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:nnx {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:nnx {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:nnx {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} { ^J \tl_trim_spaces:n {#2} } }
16 \cs_new:Nn \@@_msg_new:nnn
17 { \msg_new:nxxx {fontspec} {#1} { ^J \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; this may be but usually is not
32   a fontspec bug. Either there is a typo in the font name/file,
33   the font is not installed (correctly), or there is a bug
34   in the underlying font loading engine (XeTeX/luatflod).
35 }
36 {
37   A font might not be found for many reasons.\\
38   Check the spelling, where the font is installed etc. etc.\\
```

```

39   When in doubt, ask someone for help!
40   }
41   \@@_msg_new:nnn {rename-feature-not-exist}
42   {
43     The feature #1 doesn't appear to be defined.
44   }
45   {
46     It looks like you're trying to rename a feature that doesn't exist.
47   }
48   \@@_msg_new:nn {no-glyph}
49   {
50     '#1' does not contain glyph #2.
51   }
52   \@@_msg_new:nnn {euler-too-late}
53   {
54     The euler package must be loaded BEFORE fontspec.
55   }
56   {
57     fontspec only overwrites euler's attempt to
58     define the maths text fonts if fontspec is
59     loaded after euler. Type <return> to proceed
60     with incorrect \string\mathit, \string\mathbf, etc.
61   }
62   \@@_msg_new:nnn {no-xcolor}
63   {
64     Cannot load named colours without the xcolor package.
65   }
66   {
67     Sorry, I can't do anything to help. Instead of loading
68     the color package, use xcolor instead.
69   }
70   \@@_msg_new:nnn {unknown-color-model}
71   {
72     Error loading colour `#1'; unknown colour model.
73   }
74   {
75     Sorry, I can't do anything to help. Please report this error
76     to my developer with a minimal example that causes the problem.
77   }
78   \@@_msg_new:nnn {not-in-addfontfeatures}
79   {
80     The "#1" font feature cannot be used in \string\addfontfeatures.
81   }
82   {
83     This is due to how TeX loads fonts; such settings
84     are global so adding them mid-document within a group causes
85     confusion. You'll need to define multiple font families to achieve
86     what you want.
87   }

```

1.2 Warnings

```

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

```

```

139   '#1' doesn't appear to have an Optical Size axis.
140   }
141   \@@_msg_new:nn {language-not-exist}
142   {
143     Language '#1' not explicitly supported
144     within font '\l_fontspec_fontname_tl'
145     with script '\l_@@_script_name_tl'.
146     Check the typeset output, and if it is okay then ignore this warning.
147     Otherwise a different font should be chosen.
148   }
149   \@@_msg_new:nn {only-xetex-feature}
150   {
151     Ignored XeTeX-only feature: '#1'.
152   }
153   \@@_msg_new:nn {only-luatex-feature}
154   {
155     Ignored LuaTeX-only feature: '#1'.
156   }
157   \@@_msg_new:nn {unknown-renderer}
158   {
159     Renderer '#1' unknown. Assuming Harfbuzz with 'shaper=#1'.
160     Please raise a fontspec issue to add this shaper to the interface.
161   }
162   \@@_msg_new:nn {no-mapping}
163   {
164     Input mapping not supported in LuaTeX.
165   }
166   \@@_msg_new:nn {no-mapping-ligtex}
167   {
168     Input mapping not supported in LuaTeX.\\
169     Use "Ligatures=TeX" instead of "Mapping=tex-text".
170   }

message for package options must be loaded earlier
171 </fontspec>
172 <*options>
173 \msg_new:nnn {fontspec} {cm-default-obsolete}
174 {
175   The-"cm-default"-package-option-is-obsolete.
176 }
177 \msg_new:nnn {fontspec} {enc-obsolete}
178 {
179   The-"#1"-package-option-is-obsolete.-TU-is-the-default-encoding.
180 }
181 </options>
182 <*fontspec>
183 \@@_msg_new:nn {font-index-needs-ttc}
184 {
185   The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
186   Feature ignored.
187 }
188 \@@_msg_new:nn {feat-cannot-remove}

```



```

189 {
190 The "#1" feature cannot be deactivated. Request ignored.
191 }

```

1.3 Info messages

```

192 \@@_msg_new:nn {defining-font}
193 {
194 Font family '\g_@@_nfss_family_tl' created for font '#2'
195 with options [\l_@@_all_features_clist].\\
196 \\
197 This font family consists of the following NFSS series/shapes:\\
198 \g_@@_defined_shapes_tl
199 }
200 \@@_msg_new:nn {no-font-shape}
201 {
202 Could not resolve font "#1" (it probably doesn't exist).
203 }
204 \@@_msg_new:nn {set-scale}
205 {
206 \l_fontspeg_fontname_tl\space scale = \l_@@_scale_tl.
207 }
208 \@@_msg_new:nn {setup-math}
209 {
210 Adjusting the maths setup (use [no-math] to avoid this).
211 }
212 \@@_msg_new:nn {no-script}
213 {
214 Script '#2' not explicitly supported within font '#1'.
215 Check the typeset output, and if it is okay then ignore this warning.
216 Otherwise a different font should be chosen.
217 }
218 \@@_msg_new:nn {opa-twice}
219 {
220 Opacity set twice, in both Colour and Opacity.\\
221 Using specification "Opacity=#1".
222 }
223 \@@_msg_new:nn {opa-twice-col}
224 {
225 Opacity set twice, in both Opacity and Colour.\\
226 Using an opacity specification in hex of "#1/FF".
227 }
228 \@@_msg_new:nn {bad-colour}
229 {
230 Bad colour declaration "#1".
231 Colour must be one of:\\
232 * a named xcolor colour\\
233 * a six-digit hex colour RRGGBB\\
234 * an eight-digit hex colour RRGGBBTT with opacity
235 }

Reset 'space' behaviour:
236 \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   ,math / unknown .code:n = { } % \msg_warning:nnn {fontspec} {math-opt-unknown} {#1}
7   ,no-math .bool_gset_inverse:N = \g_@@_math_bool
8   ,no-math .usage:n = preamble
9   ,config .bool_gset:N = \g_@@_cfg_bool
10  ,config .usage:n = load
11  ,no-config .bool_gset_inverse:N = \g_@@_cfg_bool
12  ,no-config .usage:n = load
13  ,euenc .code:n = { \msg_warning:nnn {fontspec} {enc-obsolete}{euenc} }
14  ,tuenc .code:n = { \msg_warning:nnn {fontspec} {enc-obsolete}{tuenc} }
15
16  ,quiet .code:n =
17  {
18    \msg_redirect_module:nnn { fontspec } { warning } { info }
19    \msg_redirect_module:nnn { fontspec } { info } { none }
20  }
21  ,silent .code:n =
22  {
23    \msg_redirect_module:nnn { fontspec } { warning } { none }
24    \msg_redirect_module:nnn { fontspec } { info } { none }
25  }
26  ,verbose .code:n =
27  {
28    \msg_redirect_module:nnn { fontspec } { warning } { warning }
29    \msg_redirect_module:nnn { fontspec } { info } { info }
30  }
31  \msg_new:nnn {fontspec} {math-opt-unknown}
32  {
33    The~ global~ option~ 'math=#1'~ is~ not~ recognised.~ It~ will~ be~ ignored.
34  }
35
36 \SetKeys{config,math}
37 \ProcessKeyOptions
```

1.2 Encodings

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

```
38 \cs_if_exist:cF {T@TU}
```

```

39 {
40   \@@_warning:n {tu-clash}
41   \DeclareFontEncoding{TU}{-}{-}
42   \DeclareFontSubstitution{TU}{lmr}{m}{n}
43 }
44 \tl_gset:Nn \g_fontspec_encoding_tl { TU }
45 \tl_set:Nn \rmdefault {lmr}
46 \tl_set:Nn \sfdefault {lmss}
47 \tl_set:Nn \ttdefault {lmtt}
48 \RequirePackage[\g_fontspec_encoding_tl]{fontenc}
49 \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:

```
50 \tl_if_in:NnT \@filelist {.cls} { \normalsize }
```

Dealing with a couple of the problems introduced by babel:

```

51 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
52 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
53 \AtBeginDocument
54 {
55   \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
56   \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
57 }

```

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 58 \cs_new:Npn \FontspecSetCheckBoolTrue { \bool_set_true:N \l_@@_check_bool }
59 \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`

```

60 \cs_new:Nn \@@_keys_set_known:nnN
61 {
62   <debug> \typeout{::: Keys~set:~{#1}~{#2} }
63   \keys_set_known:nnN {#1} {#2} #3
64   <debug> \typeout{::: Leftover:~{#3} }
65 }
66 \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.

```

67 \cs_new:Nn \@@_int_mult_truncate:Nn
68 {
69   \int_set:Nn #1 { \__dim_eval:w #2 #1 \__dim_eval_end: }
70 }

```

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

```
\@@_lua_function:ne
\@@_lua_function:nee 71 <*LU>
\@@_lua_function:neee 72 \cs_set:Npn \@@_lua_function:ne #1#2 { \lua_now:e { fontspec.#1 ("#2")
\@@_lua_function:neeee 73 \cs_set:Npn \@@_lua_function:nee #1#2#3 { \lua_now:e { fontspec.#1 ("#2", "#3")
74 \cs_set:Npn \@@_lua_function:neee #1#2#3#4 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4")
75 \cs_set:Npn \@@_lua_function:neeee #1#2#3#4#5 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4", "#5")
76 </LU>
```

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

1.4 expl3 variants

```
77 \cs_generate_variant:Nn \int_set:Nn {Nv}
78 \cs_generate_variant:Nn \keys_set:nn {nx}
79 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
80 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
81 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
82 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
83 \cs_generate_variant:Nn \prop_gput:Nnn {NxN}
84 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
85 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
86 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
87 \cs_generate_variant:Nn \tl_if_empty_p:n {e}
88 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
89 \cs_generate_variant:Nn \tl_if_empty:nF {x}
90 \cs_generate_variant:Nn \tl_if_empty:nF {f}
91 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
92 \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:Onn { \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 <debug>\typeout{:~_main_setmainfont:nn}
27   \ifdefined\DeclareFontSeriesDefault
28     \DeclareFontSeriesDefault [rm] {bf} {\bfdefault}
29   \fi
```

```

30 \fontspec_set_family:Nnn \l_@@_rmfamily_family_tl {#1} {#2}
31 \tl_set_eq:NN \rmdefault \l_@@_rmfamily_family_tl
32 \tl_set_eq:NN \l_@@_rmfamily_encoding_tl \g_@@_nfss_enc_tl
33 \str_if_eq:eeT {\familydefault} {\rmdefault}
34 { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
35 \@@_setmainfont_hook:nn {#1} {#2} % for unicode-math only
36 \normalfont
37 }

```

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

`\setsansfont` Same as above.

```

38 \cs_new:Nn \@@_main_setsansfont:nn
39 {
40   \ifdefined\DeclareFontSeriesDefault
41     \DeclareFontSeriesDefault[sf]{bf}{\bfdefault}
42   \fi
43   \fontspec_set_family:Nnn \l_@@_sffamily_family_tl {#1} {#2}
44   \tl_set_eq:NN \sfdefault \l_@@_sffamily_family_tl
45   \tl_set_eq:NN \l_@@_sffamily_encoding_tl \g_@@_nfss_enc_tl
46   \str_if_eq:eeT {\familydefault} {\sfdefault}
47   { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
48   \@@_setsansfont_hook:nn {#1} {#2} % for unicode-math only
49   \normalfont
50 }

```

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

`\setmonofont` Same as above.

```

51 \cs_new:Nn \@@_main_setmonofont:nn
52 {
53   \ifdefined\DeclareFontSeriesDefault
54     \DeclareFontSeriesDefault[tt]{bf}{\bfdefault}
55   \fi
56   \fontspec_set_family:Nnn \l_@@_ttfamily_family_tl {#1} {#2}
57   \tl_set_eq:NN \ttdefault \l_@@_ttfamily_family_tl
58   \tl_set_eq:NN \l_@@_ttfamily_encoding_tl \g_@@_nfss_enc_tl
59   \str_if_eq:eeT {\familydefault} {\ttdefault}
60   { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
61   \@@_setmonofont_hook:nn {#1} {#2} % for unicode-math only
62   \normalfont
63 }

```

(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`.

```

64 \cs_new:Nn \@@_main_setmathrm:nn
65 {
66 (XE) \fontspec_gset_family:Nnn \g_@@_mathrm_tl {#1} {#2}
67 (LU) \fontspec_gset_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}

```

```

68     \@@_setmathrm_hook:nn {#1} {#2} % for unicode-math only
69   }

```

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

`\setboldmathrm`

```

70 \cs_new:Nn \@@_main_setboldmathrm:nn
71   {
72   \langle XE \rangle \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {#1} {#2}
73   \langle LU \rangle \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {Renderer=Basic,#1} {#2}
74     \@@_setboldmathrm_hook:nn {#1} {#2} % for unicode-math only
75   }

```

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

`\setmathsf`

```

76 \cs_new:Nn \@@_main_setmathsf:nn
77   {
78   \langle XE \rangle \fontspec_gset_family:Nnn \g_@@_mathsf_tl {#1} {#2}
79   \langle LU \rangle \fontspec_gset_family:Nnn \g_@@_mathsf_tl {Renderer=Basic,#1} {#2}
80     \@@_setmathsf_hook:nn {#1} {#2} % for unicode-math only
81   }

```

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

`\setmathtt`

```

82 \cs_new:Nn \@@_main_setmathtt:nn
83   {
84   \langle XE \rangle \fontspec_gset_family:Nnn \g_@@_mathtt_tl {#1} {#2}
85   \langle LU \rangle \fontspec_gset_family:Nnn \g_@@_mathtt_tl {Renderer=Basic,#1} {#2}
86     \@@_setmathtt_hook:nn {#1} {#2} % for unicode-math only
87   }

```

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

Hooks:

```

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

```

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

```

95 \onlypreamble\setmathrm
96 \onlypreamble\setboldmathrm
97 \onlypreamble\setmathsf
98 \onlypreamble\setmathtt

```

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

```

99 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
100 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
101 \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`).

```

102 \cs_new:Nn \@@_main_newfontfamily:NnnN
103   {
104     \fontspec_set_family:cnn { l_@@_ \cs_to_str:N #1 _family_tl } {#2} {#3}
105     \use:x
106     {
107       \exp_not:N #4 \exp_not:N #1 {}
108       {
109         \exp_not:N \fontfamily { \use:c { l_@@_ \cs_to_str:N #1 _family_tl } }
110         \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
111         \exp_not:N \selectfont
112       }
113     }
114   }

```

(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.

```

115 \cs_new:Nn \@@_main_newfontface:NnnN
116   {
117     \@@_main_newfontfamily:NnnN #1 { BoldFont={},ItalicFont={},SmallCapsFont={},#2 } {#3} #4
118   }

```

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

1.2 Font feature selection

`\@@_set_default_features:nn`

```

119 \cs_new:Nn \@@_set_default_features:nn
120   {
121     \IfBooleanTF {#1} \clist_gput_right:Nn \clist_gset:Nn
122       \g_@@_default_fontopts_clist {#2}
123   }

```

(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.

```

124 \cs_new:Nn \@@_set_font_default_features:nnn
125   {
126     <debug> \typeout{\unexpanded[_set_font_default_features:nnn:{#1}{#2}{#3}}
127     \clist_map_inline:nn {#2}
128     {
129       \tl_if_single:nTF {##1}
130         { \tl_set:No \l_@@_tmp_tl { \cs:w l_@@_ \cs_to_str:N ##1 _family_tl\cs_end: } }
131         { \@@_sanitise_fontname:Nn \l_@@_tmp_tl {##1} }
132     }

```

```

133     \IfBooleanTF {#1}
134     {
135         \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
136         { \tl_clear:N \l_@@_tmpb_tl }
137         \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
138         \prop_gput:NVV \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
139     }
140     {
141         \tl_if_empty:nTF {#3}
142         { \prop_gremove:NV \g_@@_fontopts_prop \l_@@_tmp_tl }
143         { \prop_gput:NVn \g_@@_fontopts_prop \l_@@_tmp_tl {#3,} }
144     }
145 }
146 }

```

(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.

```

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

```

```

168     }
169     \ignorespaces
170 }

```

(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.

```

171 \cs_new:Nn \@@_main_newfontfeature:nn
172 {
173     \keys_define:nn { fontspec }
174     {
175         #1 .code:n = { \@@_update_featstr:n {#2} }
176     }
177 }

```

(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.

```

178 \cs_new:Nn \@@_main_newAATfeature:nnnn
179 {
180     \keys_if_exist:nnF { fontspec } {#1}
181     { \@@_define_aat_feature_group:n {#1} }
182
183     \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
184     { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
185
186     \@@_define_aat_feature:nnnn {#1}{#2}{#3}{#4}
187 }

```

(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.

```

188 \cs_new:Nn \@@_main_newopentypefeature:nnn
189 {
190     \keys_if_exist:nnF { fontspec / options } {#1}
191     { \@@_define_opentype_feature_group:n {#1} }
192
193     \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
194     { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
195
196     \exp_args:Nnnx \@@_define_opentype_feature:nnnnn
197     {#1} {#2} { \@@_strip_plus_minus:n {#3} } {#3} {
198 }

```



```

200 \cs_new:Nn \@@_strip_plus_minus:n { \@@_strip_plus_minus_aux:Nq #1 \q_nil }
201 \cs_new:Npn \@@_strip_plus_minus_aux:Nq #1#2 \q_nil
202 {
203   \str_case:nnF {#1} { {+} {#2} {-} {#2} } {#1#2}
204 }

```

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

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

```

204 \cs_new:Nn \@@_main_aliasfontfeature:nn
205 {
206   <debug> \typeout{:::::::::::^^J:: aliasfontfeature{#1}{#2}}
207   \bool_set_false:N \l_@@_alias_bool
208
209   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
210   {
211     \keys_if_exist:nnT {##1} {#1}
212     {
213       <debug> \typeout{::: Key~exists~##1~/~#1}
214       \bool_set_true:N \l_@@_alias_bool
215       \keys_define:nn {##1}
216       { #2 .code:n = { \keys_set:nn {##1} { #1 = {####1} } } }
217     }
218   }
219
220   \bool_if:NF \l_@@_alias_bool
221   { \@@_warning:nx {rename-feature-not-exist} {#1} }
222 }

```

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

\aliasfontfeatureoption

```

223 \cs_new:Nn \@@_main_aliasfontfeatureoption:nnn
224 {
225   \bool_set_false:N \l_@@_alias_bool
226
227   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
228   {
229     \keys_if_exist:nnT { ##1 / #1 } {#2}
230     {
231       <debug> \typeout{::: Keyval~exists~##1~/~#1~-#2}
232       \bool_set_true:N \l_@@_alias_bool
233       \keys_define:nn { ##1 / #1 }
234       { #3 .code:n = { \keys_set:nn {##1} { #1 = {#2} } } }
235     }
236
237     \keys_if_exist:nnT { ##1 / #1 } {#2Reset}
238     {
239       <debug> \typeout{::: Keyval~exists~##1~/~#1~-#2Reset}
240       \keys_define:nn { ##1 / #1 }
241       { #3Reset .code:n = { \keys_set:nn {##1} { #1 = {#2Reset} } } }
242     }

```

```

243
244     \keys_if_exist:nnT { ##1 / #1 } {#20ff}
245     {
246 <debug> \typeout{::: Keyval~exists~##1~/~#1~-~#20ff}
247         \keys_define:nn { ##1 / #1 }
248             { #30ff .code:n = { \keys_set:nn {##1} { #1 = {#20ff} } } }
249     }
250 }
251
252 \bool_if:NF \l_@@_alias_bool
253 { \@@_warning:nx {rename-feature-not-exist} {#1/#2} }
254 }

```

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

\@@_main_DeclareFontExtensions:n

```

255 \cs_new:Nn \@@_main_DeclareFontExtensions:n
256 {
257     \clist_set:Nn \l_@@_extensions_clist { #1 }
258 }

```

Defaults:

```

259 \@@_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

```

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

```

```

283     {
284         \exp_args:NV \fontspec_if_current_feature:nTF \g_@@_single_feat_tl
285         { \prg_return_true: } { \prg_return_false: }
286     }
287 }

```

(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 288 \cs_new_protected:Nn \@@_main_oldstylenums:n
289 {
290     \group_begin:
291         \addfontfeature{Numbers=OldStyle}
292         #1
293     \group_end:
294 }
295 \cs_new_protected:Nn \@@_main_liningnums:n
296 {
297     \group_begin:
298         \addfontfeature{Numbers=Lining}
299         #1
300     \group_end:
301 }

```

(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.)

2 Overview

2.1 Commands

<code>\fontspec_gset_family:Nnn</code>	<code>\fontspec_set_family:Nnn</code>	<code><family> <{features}> <{font name}></code>
<code>\fontspec_set_family:Nnn</code>		

Defines a new NFSS font family from given `<features>` and ``, and stores the name in the token list variable `<family>`. See the standard fontspec user commands for applications of this function.

<code>\fontspec_gset_fontface:NNnn</code>	<code>\fontspec_set_fontface:NNnn</code>	<code><face> <family> <{features}> <{font name}></code>
<code>\fontspec_set_fontface:NNnn</code>		

As for `\fontspec_set_family:Nnn` but with a single font face only. (E.g., no bold, italic shapes, etc.) The control sequence `<face>` is a primitive T_EX font command.

2.2 Conditionals

<code>\fontspec_font_if_exist:nTF</code>	<code>\fontspec_font_if_exist:nTF</code>	<code><{font name}> Argtrue code <{false code}></code>
--	--	--

Does this font exist? The font name can refer to the 'logical' name or to a filename with known font extension.

<code>\fontspec_if_fontspec_font:TF</code>	<code>\fontspec_if_fontspec_font:TF</code>	<code><{true code}> <{false code}></code>
--	--	---

<code>\fontspec_if_aat_feature:nnTF</code>	<code>\fontspec_if_aat_feature:nnTF</code>	<code><{true code}> <{false code}></code>
--	--	---

<code>\fontspec_if_opentype:TF</code>	<code>\fontspec_if_opentype:TF</code>	<code><{true code}> <{false code}></code>
---------------------------------------	---------------------------------------	---

<u>\fontspec_if_feature:nTF</u>	\fontspec_if_feature:nTF {<feat tag>} {<>true code>} {<>false code>}	Check if the raw OpenType <feature tag> is available in the current font with script and language settings as set up when the font was loaded.
<u>\fontspec_if_feature:nnnTF</u>	\fontspec_if_feature:nnnTF {<script tag>} {<lang tag>} {<feat tag>} {<>true code>} {<>false code>}	\fontspec_if_feature:nTF {latn} {ROM} {pnum} {True} {False}
		Test whether the currently selected font with raw OpenType <script tag> and raw OpenType <language tag> contains the raw OpenType <feature tag>. Returns false if the font is not loaded by fontspec or is not an OpenType font.
<u>\fontspec_if_script:nTF</u>	\fontspec_if_script:nTF {<script tag>} {<>true code>} {<>false code>}	\fontspec_if_script:nTF {latn} {True} {False}
		Test whether the currently selected font contains the raw OpenType <script tag>. Returns false if the font is not loaded by fontspec or is not an OpenType font.
<u>\fontspec_if_language:nTF</u>	\fontspec_if_language:nTF {<lang tag>} {<>true code>} {<>false code>}	\fontspec_if_language:nTF {ROM} {True} {False}
		Check if the raw OpenType <language tag> is available in the current font with script settings as set up when the font was loaded.
<u>\fontspec_if_language:nnTF</u>	\fontspec_if_language:nnTF {<script tag>} {<lang tag>} {<>true code>} {<>false code>}	\fontspec_if_language:nnTF {cyril} {SRB} {True} {False}
		Test whether the currently selected font contains the raw OpenType <language tag> in <script tag>. Returns false if the font is not loaded by fontspec or is not an OpenType font.
<u>\fontspec_if_current_script:nTF</u>	\fontspec_if_current_script:nTF {<script tag>} {<>true code>} {<>false code>}	
		Test whether the currently loaded font has been loaded with the specified raw OpenType <script tag>.
<u>\fontspec_if_current_language:nTF</u>	\fontspec_if_current_language:nTF {<lang tag>} {<>true code>} {<>false code>}	
		Test whether the currently loaded font has been loaded with the specified raw OpenType <language tag>.
<u>\fontspec_if_current_feature:nTF</u>	\fontspec_if_current_feature:nTF {<feat tag>} {<>true code>} {<>false code>}	
		Test whether the currently loaded font is using the specified raw OpenType <feature tag>.
<u>\fontspec_if_small_caps:TF</u>	\fontspec_if_small_caps:TF {<>true code>} {<>false code>}	
		Test whether the current font has small caps available.

3 Implementation

`\fontspec_if_fontspec_font:TF`

```
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 36.)

`\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 36.)

`\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 36.)

`\fontspec_if_feature:nTF`

Test whether the currently selected font contains the raw OpenType feature #1. E.g.: `\fontspec_if_feature:nTF {pnum} {True} {False}` 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 37.)

\fontspec_if_feature:nnnTF

#1 : script tag
#2 : language tag
#3 : feature tag

```

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 37.)

\fontspec_if_script:nTF

#1 : script tag

```

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 37.)

\fontspec_if_language:nTF #1 : lang tag

```

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 37.)

\fontspec_if_language:nnTF #1 : script tag
#2 : lang tag

```

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 37.)

`\fontspec_if_current_script:nTF` #1 : script tag

```
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 37.)

`\fontspec_if_current_language:nTF` #1 : lang tag

```
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 37.)

`\fontspec_gset_family:Nnn` #1 : family

`\fontspec_set_family:Nnn` #2 : fontspec features

#3 : font

```
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   <debug>\typeout{::::::~~fontspec_set_family:Nnn}
150   \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
151   \tl_set:Nn \l_@@_family_label_tl {#1}
152   \@@_select_font_family:nn {#2} {#3}
153   \@@_tl_new_if_free:N #1
154   #4 #1 \l_fontspec_family_tl
155   \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
```

```

156 }
157 \cs_new:Nn \fontspec_gset_family:Nnn { \@@_set_family:Nnn #1 {#2} {#3} \tl_gset_eq:NN }
158 \cs_new:Nn \fontspec_set_family:Nnn { \@@_set_family:Nnn #1 {#2} {#3} \tl_set_eq:NN }
159 \cs_generate_variant:Nn \fontspec_set_family:Nnn {c}

```

(End of definition for `\fontspec_gset_family:Nnn` and `\fontspec_set_family:Nnn`. These functions are documented on page 36.)

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

```

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

```

(End of definition for `\fontspec_gset_fontface:NNnn` and `\fontspec_set_fontface:NNnn`. These functions are documented on page 36.)

`\fontspec_font_if_exist:nTF`

```

172 \prg_new_conditional:Nnn \fontspec_font_if_exist:n {TF,T,F}
173 {
174   \group_begin:
175     \@@_init:
176     \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
177     \@@_primitive_font_if_exist:nTF { \@@_construct_font_call:nn {#1} {} }
178     { \group_end: \prg_return_true: }
179     { \group_end: \prg_return_false: }
180 }
181 \cs_set_eq:NN \IfFontExistsTF \fontspec_font_if_exist:nTF

```

(End of definition for `\fontspec_font_if_exist:nTF`. This function is documented on page 36.)

`\fontspec_if_current_feature:nTF` #1 : feat tag

```

182 \prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
183 {
184   <debug>\typeout{::~fontspec_if_current_feature:n~{#1}}
185   <debug>\typeout{::::~primitive_font_current_name::~~\@@_primitive_font_current_name:}
186   \exp_args:Nxx \tl_if_in:nnTF
187   { \@@_primitive_font_current_name: } { \tl_to_str:n {#1} }
188   { \prg_return_true: } { \prg_return_false: }
189 }

```

(End of definition for `\fontspec_if_current_feature:nTF`. This function is documented on page 37.)

`\fontspec_if_small_caps:TF`

```
190 \prg_new_conditional:Nnn \fontspec_if_small_caps: {TF,T,F}
191 {
192   \@@_if_merge_shape:nTF {sc}
193   {
194     \tl_set_eq:Nc \l_@@_smcp_shape_tl { \@@_shape_merge:nn {\f@shape} {sc} }
195   }
196   {
197     \tl_set:Nn \l_@@_smcp_shape_tl {sc}
198   }
199
200   \cs_if_exist:cTF { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
201   {
202     \tl_if_eq:ccTF
203     { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
204     { \f@encoding/\f@family/\f@series/\shapedefault }
205     { \prg_return_false: }
206     { \prg_return_true: }
207   }
208   { \prg_return_false: }
209 }
```

(End of definition for `\fontspec_if_small_caps:TF`. This function is documented on page 37.)

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 <debug>\typeout{^^J::::::::::::::::::::: l_fontspec_fontname_tl~ =~ \l_fontspec_fontname_tl }
13 <debug>\typeout{::::::::::::::::::::: _fontname_up_tl~ =~ \l_@@_fontname_up_tl }
14 <debug>\typeout{::::::::::::::::::::: l_@@_extension_tl~ =~ \l_@@_extension_tl }
15
16   \@@_if_detect_external:nT {#2}
17     { \keys_set:nn {fontspec-prepare-external} {Path} }
18
19   \keys_set_known:nn {fontspec-prepare-cfg} {#1}
20
21   \@@_init_ttc:n {#2}
22   \@@_load_external_fontoptions:N \l_fontspec_fontname_tl
23
```

```

24     \@@_extract_all_features:n {#1}
25     \tl_set:Nx \l_@@_fontid_tl { \tl_to_str:N \l_fontspec_fontname_tl--\tl_to_str:N \l_@@_all
26
27 <debug>\typeout{fontid: \l_@@_fontid_tl}
28
29     \@@_preparse_features:
30
31 <debug>\typeout{^^J:~::~::~::::: l_fontspec_fontname_tl~ =~ \l_fontspec_fontname_tl }
32 <debug>\typeout{~::~::::: _fontname_up_tl~ =~ \l_@@_fontname_up_tl }
33 <debug>\typeout{~::~::::: l_@@_extension_tl~ =~ \l_@@_extension_tl }
34
35     \@@_load_font:
36     \@@_set_scriptlang:
37     \@@_get_features:n {}
38     \bool_set_false:N \l_@@_firsttime_bool
39
40     \@@_save_family_needed:nTF {#2}
41     {
42         \@@_save_family:nn {#1} {#2}
43 <debug>\@@_warning:nxx {defining-font} {#1} {#2}
44     }
45     {
46 <debug>\typeout{Font~ family~ already~ defined.}
47     }
48     \group_end:
49
50     \tl_set_eq:NN \l_fontspec_family_tl \g_@@_nfss_family_tl
51 }

```

(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:

```

52 \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.

```

53 \cs_new:Nn \@@_sanitise_fontname:Nn
54 {
55     \tl_set:Nx #1 {#2}
56     \tl_trim_spaces:N #1
57     \@@_process_ext:N #1
58 }
59
60 \cs_new:Nn \@@_process_ext:N
61 {
62     \clist_map_inline:Nn \l_@@_extensions_clist
63     {
64         \tl_if_in:NnT #1 {##1}
65         {
66             \tl_remove_once:Nn #1 {##1}

```

```

67         \tl_set:Nn \l_@@_extension_tl {##1}
68         \@@_font_is_file:
69         \clist_map_break:
70     }
71 }
72 }

```

(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.

```

73 \prg_new_conditional:Nnn \@@_if_detect_external:n {T}
74 {
75 <debug> \typeout{:: @@_if_detect_external:n { \exp_not:n {#1} } }
76 \clist_map_inline:Nn \l_@@_extensions_clist
77 {
78     \bool_set_false:N \l_@@_tmpa_bool
79     \exp_args:Nx % <- this should be handled earlier
80     \tl_if_in:nnT {#1 <= end_of_string} {##1 <= end_of_string}
81     { \bool_set_true:N \l_@@_tmpa_bool \clist_map_break: }
82 }
83 \bool_if:NTF \l_@@_tmpa_bool \prg_return_true: \prg_return_false:
84 }

```

(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.

```

85 \cs_new:Nn \@@_init_ttc:n
86 {
87     \str_if_eq:eeT { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
88     {
89         \@@_sanitise_fontname:Nn \l_@@_fontname_it_tl {#1}
90         \@@_sanitise_fontname:Nn \l_@@_fontname_bf_tl {#1}
91         \@@_sanitise_fontname:Nn \l_@@_fontname_bfit_tl {#1}
92     }
93 }

```

(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 token list containing a sanitised fontname. In the past this used a space-stripped version of the name, so we check for the file both with and without spaces to load it.

```

94 \cs_new:Nn \@@_load_external_fontoptions:N
95 {
96     \bool_if:NT \l_@@_fontcfg_bool
97     {
98 <debug> \typeout{:: @@_load_external_fontoptions:N \exp_not:N #1 }
99         \tl_set:Nx \l_@@_ext_filename_tl {#1.fontspec}
100         \tl_remove_all:Nn \l_@@_ext_filename_tl {-}
101         \prop_if_in:NVF \g_@@_fontopts_prop #1
102         {

```

```

103     \exp_args:No \file_if_exist:nTF { \l_@@_ext_filename_tl }
104     {
105         \file_input:n { \l_@@_ext_filename_tl }
106     }
107     {
108         \tl_remove_all:Nn \l_@@_ext_filename_tl {~}
109         \exp_args:No \file_if_exist:nTF { \l_@@_ext_filename_tl }
110         { \file_input:n { \l_@@_ext_filename_tl } }
111     }
112 }
113 }
114 }

```

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

\@@_extract_all_features:

```

115 \cs_new:Nn \@@_extract_all_features:n
116 {
117 <debug> \typeout{:: @@_extract_all_features:n { \unexpanded {#1} } }
118 \bool_if:NTF \l_@@_disable_defaults_bool
119 {
120     \clist_set:Nx \l_@@_all_features_clist {#1}
121 }
122 {
123     \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
124     { \clist_clear:N \l_@@_fontopts_clist }
125
126     \prop_get:NVNF \g_@@_fontopts_prop \l_@@_family_label_tl \l_@@_family_fontopts_clist
127     { \clist_clear:N \l_@@_family_fontopts_clist }
128     \tl_clear:N \l_@@_family_label_tl
129
130     \clist_set:Nx \l_@@_all_features_clist
131     {
132         \g_@@_default_fontopts_clist,
133         \l_@@_family_fontopts_clist,
134         \l_@@_fontopts_clist,
135         #1
136     }
137 }
138 }

```

(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.

```

139 \cs_new:Nn \@@_preparse_features:
140 {
141 <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.

```

142
143   \l_keys_set_known:nxN {fontspec-preparse-external}
144     { \l_@@_all_features_clist }
145     \l_@@_keys_leftover_clist
146

```

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.

```

147   \tl_set_eq:NN \l_fontspec_fontname_tl \l_@@_fontname_up_tl
148   \l_keys_set_known:nxN {fontspec-renderer} {\l_@@_keys_leftover_clist}
149     \l_@@_keys_leftover_clist
150   \l_keys_set_known:nxN {fontspec-preparse} {\l_@@_keys_leftover_clist}
151     \l_@@_fontfeat_clist
152 }

```

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

`\l_load_font:`

```

153 \cs_new:Nn \l_load_font:
154 {
155   <debug>\typeout{: : \l_load_font}
156
157   <debug>\typeout{Set~ base~ font~ for~ preliminary~ analysis: \l_construct_font_call:nn { \l_@@_
158     \l_primitive_font_set:NnnF \l_@@_test_font
159     { \l_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
160     { \f@size pt - 2sp }
161     { \l_error:nx {font-not-found} {\l_@@_fontname_up_tl} }
162
163   <debug>\typeout{Set~ base~ font~ properly: \l_construct_font_call:nn { \l_@@_fontname_up_tl }
164     \l_set_font_type:N \l_@@_test_font
165     \l_primitive_font_gset:Onn \l_@@_fontface_cs_tl
166     { \l_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
167     { \f@size pt + 2sp }
168
169     \l_@@_fontface_cs_tl % this is necessary for LuaLaTeX to check the scripts properly
170
171 }

```

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

`\l_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.

```

172 \cs_new:Nn \l_construct_font_call:nnnnnn

```



```

173 {
174 <XE> " \@@_fontname_wrap:n { #1 #2 #3 }
175 <LU> " \@@_fontname_wrap:n { #1 #2 } #3
176 #4 #5
177 \str_if_eq:eeF {#6}{ } {:#6} "
178 }

```

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

```

179 \cs_new:Nn \@@_construct_font_call:nn
180 {
181   \@@_construct_font_call:nnnnnn
182   {#1}
183   \l_@@_extension_tl
184   \l_@@_ttc_index_tl
185   \l_@@_renderer_tl
186   \l_@@_optical_size_tl
187   {#2}
188 }

```

(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.

```

189 \cs_new:Nn \@@_font_is_name:
190 {
191 <XE> \cs_set_eq:NN \@@_fontname_wrap:n \use:n
192 <LU> \cs_set:Npn \@@_fontname_wrap:n ##1 { name: ##1 }
193 }

```

```

194 \cs_new:Nn \@@_font_is_file:
195 {
196 <debug> \typeout{:: _font_is_file:}
197   \bool_set_true:N \l_@@_external_bool
198   \bool_lazy_and:nnTF { \l_@@_external_kpse_bool } { \tl_if_empty_p:N \l_@@_font_path_tl }
199   {
200     \cs_set:Npn \@@_fontname_wrap:n ##1 { kpse: ##1 }
201   }
202   {
203     \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
204   }
205 }

```

(End of definition for \@@_font_is_file: and \@@_font_is_name:. 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.

```

206 \cs_new:Nn \@@_set_scriptlang:
207 {
208 <debug> \typeout{:: _set_scriptlang:}
209   \bool_if:NT \l_@@_firsttime_bool

```

```

210     {
211         \tl_if_empty:NF \l_@@_script_name_tl
212         {
213 <debug> \typeout{::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
214         \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
215         \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
216         }
217     }
218 }

```

(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.

```

219 \cs_new:Nn \@@_get_features:n
220 {
221 <debug> \typeout{:: @@@_get_features:Nn { \exp_not:n {#1} } }
222 \@@_init_fontface:
223 \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#1}
224 \l_@@_keys_leftover_clist
225 \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
226 <*XE>
227 \bool_if:NTF \l_@@_ot_bool
228 {
229 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
230 \keys_set_known:nV {fontspec-opentype} \l_@@_keys_leftover_clist
231 }
232 {
233 <debug> \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features:~"\l_@@_keys_leftover_c
234 \bool_if:NTF { \l_@@_atsui_bool || \l_@@_graphite_bool }
235 { \keys_set_known:nV {fontspec-aat} \l_@@_keys_leftover_clist }
236 }
237 </XE>
238 <*LU>
239 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
240 \keys_set_known:nV {fontspec-opentype} \l_@@_keys_leftover_clist
241 </LU>
242
243 \tl_if_empty:NF \l_@@_mapping_tl
244 { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
245
246 \str_if_eq:eeF { \l_@@_hexcol_tl \l_@@_opacity_tl }
247 { \c_@@_hexcol_tl \c_@@_opacity_tl }
248 <XE> { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
249 <LU> { \@@_update_featstr:n { color = {\l_@@_hexcol_tl\l_@@_opacity_tl} } }
250 }

```

(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.

```

251 \prg_new_conditional:Nnn \@@_save_family_needed:n { TF }
252   {
253
254   <debug> \typeout{save~ family:~ #1}
255   <debug> \typeout{== fontid_tl: "\l_@@_fontid_tl".}
256
257   \tl_if_empty:NTF \l_@@_nfss_fam_tl
258     {
259     \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
260     {
261     \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
262     \prg_return_false:
263     }
264     {
265     \tl_set:Nx \l_@@_tmp_tl {#1}
266     \tl_remove_all:Nn \l_@@_tmp_tl { ~ }
267     \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl
268     \prg_return_true:
269     }
270   }
271   {
272   \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
273   \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl _prop }
274   \prg_return_true:
275   }
276 }
277 \cs_new:Nn \@@_save_fontid_family:nn
278   {
279   \prop_get:NnNTF \g_@@_family_int_prop {#2} \l_@@_tmp_tl
280   {
281   \tl_set:Nx \l_@@_tmp_tl
282     { \int_eval:n { \l_@@_tmp_tl + 1 } }
283   }
284   { \tl_set:Nn \l_@@_tmp_tl { 0 } }
285   \prop_gput:NnV \g_@@_family_int_prop {#2} \l_@@_tmp_tl
286   \tl_gset:Nx \g_@@_nfss_family_tl { #2 ( \l_@@_tmp_tl ) }
287   \prop_gput:NnV \g_@@_fontid_family_prop {#1} \g_@@_nfss_family_tl
288   }
289 \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.

```

290 \cs_new:Nn \@@_save_family:nn
291   {
292   \@@_save_fontinfo:n {#2}

```

```

293   \@@_find_autofonts:
294   \DeclareFontFamily{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{
295   \@@_set_faces:
296   \@@_info:nxx {defining-font} {#1} {#2}
297   }

```

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

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

```

298 \cs_new:Nn \@@_save_fontinfo:n
299 {
300   \prop_new:c {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop}
301   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
302   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options} { \l_@@_all_features }
303   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
304   {
305     \@@_construct_font_call:nn {\l_fontspeg_fontname_tl}
306     { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist \@@_get_variations: }
307   }
308   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
309   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num} \l_@@_language_int
310   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_@@_script_tl
311   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_@@_lang_tl
312 }

```

(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:

```

313 \cs_new:Nn \@@_find_autofonts:
314 {
315   \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
316   {
317     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
318     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
319     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspeg_fontname_tl} {/BI}
320   }
321
322   \bool_if:NF \l_@@_nobf_bool
323   {
324     \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspeg_fontname_tl} {/B}
325   }

```

```

326
327 \bool_if:NF \l_@@_noit_bool
328 {
329   \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspeg_fontname_tl} {/I}
330 }
331
332 \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
333 }

```

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

\@@_set_faces:

```

334 \cs_new:Nn \@@_set_faces:
335 {
336   \@@_add_nfssfont:nmmm \mddefault \shapedefault \l_fontspeg_fontname_tl \l_@@_fontfeat_up_c
337   \@@_add_nfssfont:nmmm \bfdefault \shapedefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_c
338   \@@_add_nfssfont:nmmm \mddefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_c
339   \@@_add_nfssfont:nmmm \mddefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_c
340   \@@_add_nfssfont:nmmm \mddefault \swdefault \l_@@_fontname_sw_tl \l_@@_fontfeat_sw_c
341   \@@_add_nfssfont:nmmm \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit
342   \@@_add_nfssfont:nmmm \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl
343   \@@_add_nfssfont:nmmm \bfdefault \swdefault \l_@@_fontname_bfsw_tl \l_@@_fontfeat_bfsw
344   \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nmmmm #2 }
345 }
346 \cs_new:Nn \@@_set_faces_aux:nmmmm
347 {
348   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
349   \@@_make_font_shapes:Nmmmm \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
350 }

```

(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”).

```

351 \cs_new:Nn \fontspec_complete_fontname:Nn
352 {
353   \tl_set:Nx #1 {#2}
354   \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
355   \@@_process_ext:N #1
356 }

```

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

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

```

357 \cs_new:Nn \@@_add_nfssfont:nmmm
358 {
359   \tl_set:Nx \l_@@_this_font_tl {#3}

```

```

360
361 \tl_if_empty:xTF {#4}
362 { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
363 { \@@_keys_set_known:nxN {fontspec-preparse-nested} {#4} \l_@@_tmp_tl }
364
365 \tl_if_empty:NF \l_@@_this_font_tl
366 {
367   \prop_put:Nxx \l_@@_nfssfont_prop {#1/#2}
368   { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
369 }
370 }

```

(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 `ATSUI` 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_fontspeg_test_font` is an `AAT` font or an `OpenType` font or a font with feature axes (either `AAT` or `Multiple Master`), respectively.

```

371 \cs_new:Nn \@@_set_font_type:N
372 {
373   <debug> \typeout{:: @@@_set_font_type:}
374   <*XE>
375   \bool_set_false:N \l_@@_tfm_bool
376   \bool_set_false:N \l_@@_atsui_bool
377   \bool_set_false:N \l_@@_ot_bool
378   \bool_set_false:N \l_@@_mm_bool
379   \bool_set_false:N \l_@@_graphite_bool
380   \ifcase\XeTeXfonttype #1
381   <debug> \typeout{::: TFM}
382   \bool_set_true:N \l_@@_tfm_bool
383   \or
384   <debug> \typeout{::: AAT}
385   \bool_set_true:N \l_@@_atsui_bool
386   \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/AAT} }
387   \ifnum\XeTeXcountvariations #1 > 0\relax
388   <debug> \typeout{::: MM}
389   \bool_set_true:N \l_@@_mm_bool
390   \fi
391   \or
392   <debug> \typeout{::: OpenType}
393   \bool_set_true:N \l_@@_ot_bool
394   \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/OT} }
395   \or
396   <debug> \typeout{::: Graphite}
397   \bool_set_true:N \l_@@_graphite_bool
398   \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/GR} }
399   \fi
400   </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:

```

401 <*LU>
402   \bool_set_true:N \l_@@_ot_bool
403 </LU>
404   }

```

(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 \XeTeX anyway; todo: test with LuaTeX). If `` 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 `` is empty, then proceed. If not found, `` remains empty. Otherwise, we have a match.

```

405 \cs_new:Nn \@@_set_autofont:Nnn
406   {
407     \bool_if:NF \l_@@_external_bool
408     {
409       \tl_if_empty:xF {#2}
410       {
411         \tl_if_empty:NT #1
412         {
413           \@@_if_autofont:nnTF {#2} {#3}
414           { \tl_set:Nx #1 {#2#3} }
415           { \@@_info:nx {no-font-shape} {#2#3} }
416         }
417       }
418     }
419   }

```

```

420 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
421   {
422     \group_begin:
423     \@@_primitive_font_set:Nnn \l_@@_tmpa_font { \@@_construct_font_call:nn {#1} { \l_@@_pre
424     \@@_primitive_font_set:Nnn \l_@@_tmpb_font { \@@_construct_font_call:nn {#1#2} { \l_@@_pre
425     \cs_if_eq:NNTF \l_@@_tmpa_font \l_@@_tmpb_font
426     { \group_end: \prg_return_false: }
427     { \group_end: \prg_return_true: }
428   }

```

(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.
429 \cs_new:Nn \@@_make_font_shapes:Nnnnn
430   {
431     \group_begin:
432     \@@_keys_set_known:nxN {fontspec-preparse-external} { #4 } \l_@@_leftover_clist
433     \@@_load_fontname:Nn \l_fontspeg_fontname_tl {#1}
434     \@@_declare_shape:nxxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
435     \group_end:
436   }

437 \cs_new:Nn \@@_load_fontname:Nn
438   {
439   <debug> \typeout{: : \@@_load_fontname:Nn \exp_not:N #1 (#1) {#2} }
440     \@@_sanitise_fontname:Nn #1 {#2}
441     \@@_load_external_fontoptions:N #1
442     \prop_get:NVNF \g_@@_fontopts_prop #1 \l_@@_fontopts_clist
443     { \clist_clear:N \l_@@_fontopts_clist }
444     \keys_set_groups:nnV {fontspec/fontname} {getfontname} \l_@@_fontopts_clist
445     \@@_primitive_font_set:OnnF \l_@@_fontface_cs_tl
446     { \@@_construct_font_call:nn {#1} { \l_@@_pre_feat_sclist } } { \f@size pt + 2sp }
447     { \@@_error:nx {font-not-found} {#2} }
448   }

449 \keys_define:nn {fontspec/fontname}
450   {
451     Font .tl_set:N = \l_fontspeg_fontname_tl ,
452     Font .groups:n = {getfontname} ,
453   }

```

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

```

\@@_declare_shape:nxxx #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>}}.

```

454 \cs_new:Nn \@@_declare_shape:nxxx
455   {
456   <debug>\typeout{=~ declare_shape:~{\l_fontspeg_fontname_tl}~{#1}~{#2}}
457     \tl_build_begin:N \l_@@_nfss_tl
458     \tl_build_begin:N \l_@@_nfss_sc_tl
459     \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspeg_fontname_tl

```



```

460
461 \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
462
463 \tl_build_end:N \l_@@_nfss_tl
464 \tl_build_end:N \l_@@_nfss_sc_tl
465
466 \@@_declare_shapes_normal:nn {#1} {#2}
467 \@@_declare_shapes_smcaps:nn {#1} {#2}
468 \@@_declare_shape_slanted:nn {#1} {#2}
469 \@@_declare_shapes_bx:nn {#1} {#2}
470 \@@_declare_shape_loginfo:nn {#1} {#2}
471 }
472 \cs_generate_variant:Nn \@@_declare_shape:nmmm {nnxx}

```

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

\@@_setup_single_size:nn

```

473 \cs_new:Nn \@@_setup_single_size:nn
474 {
475   \tl_clear:N \l_@@_size_tl
476   \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
477
478   \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
479   \l_@@_sizing_leftover_clist
480   \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
481 <debug>\typeout{==~ size:~\l_@@_size_tl}
482
483   % "normal"
484   \@@_load_fontname:Nn \l_fontspec_fontname_tl {\l_@@_sizedfont_tl}
485   \@@_setup_nfss:Nmm \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
486 <debug> \typeout{===~ sized~ font:~ \l_@@_sizedfont_tl}
487
488   % small caps
489   \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist
490
491   \bool_if:NF \l_@@_nosc_bool
492   {
493     \tl_if_empty:NTF \l_@@_fontname_sc_tl
494     {
495       \@@_make_smallcaps:TF
496       {
497 <debug>\typeout{====~Small~ caps~ found.}
498         \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
499       }
500     }
501 <debug>\typeout{====~Small~ caps~ not~ found.}
502     \bool_set_true:N \l_@@_nosc_bool
503   }
504 }
505 { \@@_load_fontname:Nn \l_fontspec_fontname_tl {\l_@@_fontname_sc_tl} }% local for e
506 }

```

```

507
508     \bool_if:NF \l_@@_nosc_bool
509     {
510         \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
511         {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
512     }
513 }

```

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

\@@_setup_nfss:Nnnn

```

514 \cs_new:Nn \@@_setup_nfss:Nnnn
515 {
516     <debug>\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspeg_fontname_tl}
517
518     \@@_get_features:n { #2 , #3 , #4 }
519     <debug>\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist \@@_get_variations:}
520
521     \tl_if_empty:NF \l_@@_scale_tl
522     {
523         \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
524     }
525
526     \tl_build_put_right:Nx #1
527     {
528         <\l_@@_size_tl> \l_@@_scale_tl
529         \@@_construct_font_call:nn { \l_fontspeg_fontname_tl }
530         { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist \@@_get_variations: }
531     }
532 }

```

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

\@@_declare_shapes_normal:nn

```

533 \cs_new:Nn \@@_declare_shapes_normal:nn
534 {
535     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
536     {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
537 }

```

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

\@@_declare_shapes_smcaps:nn

```

538 \cs_new:Nn \@@_declare_shapes_smcaps:nn
539 {
540     \tl_if_empty:NF \l_@@_nfss_sc_tl
541     {
542         \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl} {#1}
543         { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
544     }
545 }

```

```

546 \cs_new:Nn \@@_combo_sc_shape:n
547 {
548   \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
549     { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
550     { \scdefault#1 }
551 }

```

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

\@@_DeclareFontShape:nnnnnn

```

552 \cs_new:Nn \@@_DeclareFontShape:nnnnnn
553 {
554   <debug>\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
555   \group_begin:
556   \normalsize
557   \cs_undefine:c {#1/#2/#3/#4/\f@size}
558   \group_end:
559   \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
560 }
561 \cs_generate_variant:Nn \@@_DeclareFontShape:nnnnnn {xxxxxx}

```

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.

```

562 \cs_new:Nn \@@_declare_shape_slanted:nn
563 {
564   \bool_if:nT
565     {
566       \str_if_eq_p:ee {#2} {\itdefault} &&
567       !(\str_if_eq_p:ee {\itdefault} {\sldefault})
568     }
569     {
570       \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}
571       {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
572     }
573 }

```

Similar processing for setting up b/bx substitutions.

```

\@@_declare_shapes_bx:nn 574 \cs_new:Nn \@@_declare_shapes_bx:nn
575 {
576   \bool_if:nT
577     {
578       \str_if_eq_p:ee {#1} {\bfdefault} &&
579       !(\str_if_eq_p:ee {\bfdefault} {bx})
580     }
581     {
582       % bx/?
583       \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}

```

```

584     {bx} {#2}
585     { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
586     { \l_@@_postadjust_tl }
587
588 % bx/sc -> b/sc
589 \tl_if_empty:NF \l_@@_nfss_sc_tl
590 {
591     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
592     {bx} { \@@_combo_sc_shape:n {#2} }
593     { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
594     { \l_@@_postadjust_tl }
595 }
596
597 % bx/sl -> bx/it
598 \bool_if:nT
599 {
600     \str_if_eq_p:ee {#2} {\itdefault} &&
601     !(\str_if_eq_p:ee {\itdefault} {\sldefault})
602 }
603 {
604     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
605     {bx} {\sldefault}
606     { <->ssub*\g_@@_nfss_family_tl/bx/\itdefault }
607     { \l_@@_postadjust_tl }
608 }
609
610 }
611 }

```

Lastly some informative messaging.

```

\@@_declare_shape_loginfo:nn 612 \cs_new:Nn \@@_declare_shape_loginfo:nn
613 {
614     \tl_gput_right:Nx \g_@@_defined_shapes_tl
615     {
616         \exp_not:n { \ }
617         -- \exp_not:N \str_case:nn {#1/#2}
618         {
619             {\mddefault/\shapedefault} {'normal'~}
620             {\bfdefault/\shapedefault} {'bold'~}
621             {\mddefault/\itdefault} {'italic'~}
622             {\mddefault/\sldefault} {'slanted'~}
623             {\mddefault/\swdefault} {'swash'~}
624             {\bfdefault/\itdefault} {'bold~ italic'~}
625             {\bfdefault/\sldefault} {'bold~ slanted'~}
626             {\bfdefault/\swdefault} {'bold~ swash'~}
627         } (#1/#2)~
628         with~ NFSS~ spec.:~
629         \l_@@_nfss_tl
630         \exp_not:n { \ }
631         -- \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
632         {
633             {\mddefault/\scdefault} {'small~ caps'~}

```

```

634     {\bfdefault/\scdefault} {'bold~ small~ caps'~}
635     {\mddefault/\scitdefault} {'italic~ small~ caps'~}
636     {\bfdefault/\scitdefault} {'bold~ italic~ small~ caps'~}
637     {\mddefault/\scsldefault} {'slanted~ small~ caps'~}
638     {\bfdefault/\scsldefault} {'bold~ slanted~ small~ caps'~}
639 }~( #1 / \@@_combo_sc_shape:n {#2} )~
640 with~ NFSS~ spec.:~
641 \l_@@_nfss_sc_tl
642 \tl_if_empty:fF {\l_@@_postadjust_tl}
643 {
644     \exp_not:N \ and~ font~ adjustment~ code:
645     \exp_not:N \ \l_@@_postadjust_tl
646 }
647 }
648 }

```

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 649 \tl_set:Nn \l_@@_pre_feat_sclist
650 <*XE>
651 {
652     \bool_if:NT \l_@@_ot_bool
653     {
654         \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }
655         \tl_if_empty:NF \l_@@_lang_tl { language = \l_@@_lang_tl ; }
656     }
657 }
658 </XE>
659 <*LU>
660 {
661     mode = \l_@@_mode_tl ;
662     \tl_if_empty:NF \l_@@_shaper_tl { shaper = \l_@@_shaper_tl ; }
663     \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }
664     \tl_if_empty:NF \l_@@_lang_tl { language = \l_@@_lang_tl ; }
665 }
666 </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF 667 <LU>\cs_new:Nn \@@_make_smallcaps:TF
668 <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
669 {
670     \exp_args:No \@@_check_ot_feat:NnTF \l_@@_fontface_cs_tl {smcp} {#1} {#2}
671 }
672 <*XE>
673 \cs_new:Nn \@@_make_smallcaps:TF
674 {
675     \bool_if:NTF \l_@@_ot_bool
676     { \@@_make_ot_smallcaps:TF {#1} {#2} }
677     {

```

```

678     \bool_if:NT \l_@@_atsui_bool
679     {
680         \exp_args:No \@@_make_AAT_feature_string:NnnTF
681         \l_@@_fontface_cs_tl {3} {3} {#1} {#2}
682     }
683 }
684 }
685 </XE>

```

\g_@@_rawfeatures_sclist is the string used to define the list of specific font features.

\@@_update_featstr:n Each time another font feature is requested, this macro is used to add that feature to the list. Font features are separated by semicolons.

```

686 \cs_new:Nn \@@_update_featstr:n
687 {
688 <debug>     \typeout{::: @@_update_featstr:n {#1}}
689     \bool_if:NF \l_@@_firsttime_bool
690     {
691         \tl_gset:Nx \g_@@_single_feat_tl { #1 }
692 <debug>     \typeout{:::~ Adding~ feature.}
693         \tl_gput_right:Nx \g_@@_rawfeatures_sclist {#1};
694     }
695 }

```

```

\@@_remove_clashing_featstr:n 696 \cs_new:Nn \@@_remove_clashing_featstr:n
697 {
698 <debug>     \typeout{::: @@_remove_clashing_featstr:n {#1}}
699     \clist_map_inline:nn {#1}
700     {
701 <debug>     \typeout{:::~ Removing~ feature~ "##1;}
702         \tl_gremove_all:Nn \g_@@_rawfeatures_sclist {##1};
703     }
704 }
705 \cs_generate_variant:Nn \@@_remove_clashing_featstr:n {x}

```

\@@_get_variations: builds the feature string representing the current variation in-

\@@_get_variations: stance and/or axis settings.

```

706 \cs_generate_variant:Nn \tl_tail:n { e }
707 \cs_new:Nn \@@_format_axis:nn
708 {
709     , #1 = #2
710 }
711 \cs_new:Nn \@@_get_variations:
712 {
713     \tl_if_empty:NF \g_@@_instance_tl
714     {
715         instance = { \g_@@_instance_tl };
716     }
717     \prop_if_empty:NF \g_@@_rawvariations_prop
718     {
719         axis = {

```

```

720         \tl_tail:e {
721             \prop_map_function:NN \g_@@_rawvariations_prop \@@_format_axis:nn
722         }
723     };
724 }
725 }

```

1.3 Initialisation

Initialisations that need to occur once per fontspec font invocation. (Some of these may be redundant. Check whether they're assigned to globally or not.)

```

726 \cs_set:Npn \@@_init:
727 {
728 <debug> \typeout{: @@@_init:}
729     \bool_set_false:N \l_@@_ot_bool
730     \bool_set_true:N \l_@@_firsttime_bool
731     \@@_font_is_name:
732     \tl_clear:N \l_@@_font_path_tl
733     \tl_clear:N \l_@@_optical_size_tl
734     \tl_clear:N \l_@@_ttc_index_tl
735     \tl_clear:N \l_@@_renderer_tl
736     \tl_gclear:N \g_@@_defined_shapes_tl
737     \tl_gclear:N \g_@@_curr_series_tl
738     \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspeg_encoding_tl
739 <*LU>
740     \tl_set:Nn \l_@@_mode_tl {node}
741 </LU>
742 }

```

Executed in \@@_get_features:Nn.

```

\@@_init_fontface: 743 \cs_new:Nn \@@_init_fontface:
744 {
745     \tl_gclear:N \g_@@_rawfeatures_sclist
746     \prop_gclear:N \g_@@_rawvariations_prop
747     \tl_gclear:N \g_@@_instance_tl
748     \tl_clear:N \l_@@_scale_tl
749     \tl_set_eq:NN \l_@@_opacity_tl \c_@@_opacity_tl
750     \tl_set_eq:NN \l_@@_hexcol_tl \c_@@_hexcol_tl
751     \tl_set_eq:NN \l_@@_postadjust_tl \c_@@_postadjust_tl
752     \tl_clear:N \l_@@_wordspace_adjust_tl
753     \tl_clear:N \l_@@_punctspace_adjust_tl
754 }

```

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

```

\@@_ot_validate_tag:n 755 <*LU>
756 \cs_new_protected:Nn \@@_ot_validate_tag:n
757 {
758     \@@_ot_validate_tag:w #1 \q_nil

```

```

759 }
760 \cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
761 \cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
762 {
763   \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
764     { \@@_ot_validate_tag_aux:w #2 \c_empty_tl \c_empty_tl \q_nil }
765     { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
766 }
767 \cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
768 {
769   \int_compare:nT { \tl_count:n {#5} > 2 }
770     { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
771 }
772 </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 Lua_TEX, as the checking for that engine is done via Lua code provided by luaotfload.

```

773 < *XE >
774 \cs_new:Nn \@@_iv_str_to_num:Nn
775 {
776   < debug > \typeout{ \iv_str_to_num:~#1~/~#2 }
777   \@@_strip_leading_sign:Nw #1#2 \q_nil
778 }
779 \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 \c_empty_tl s, and anything beyond four chars is snipped. The \c_empty_tl s then are used to reconstruct the spaces in the string to number calculation.

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

```

780 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
781 {
782   \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
783     { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
784     { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
785 }

```

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

```

786 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
787 {
788   \int_compare:nT { \tl_count:n {#6} > 2 }
789     { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }
790
791   \int_set:Nn #1
792     {

```



```
793         `#2 * "1000000
794     + `#3 * "10000
795     + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
796     + \ifx \c_empty_tl #5 32 \else `#5 \fi
797     }
798 }
799 </XE>
```

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:nmnnn #2 : Feature option val
#3 : Check feature — leave empty for no check
#4 : Exact tag string to activate — leave empty for disable only
#5 : Tags to remove (clist)

16 \cs_new:Nn \@@_feat_prop_add:nn
17 {
18   \tl_if_empty:nF {#1}
19     {
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:nmnnn
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:nmnn

```

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:nmnnn {#1} {#2} {#3} {+#4} {#5}
55     \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#1} { \@@_feat_off:n {#2} } {#3} {-#4}
56     \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#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:nmnn

```

58 \cs_new:Nn \@@_define_opentype_onreset:nmnnn
59 {
60     \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#1} {#2} {#3} {#4} {#5}
61     \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#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,F}
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:NNTF \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:NNTF \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. `\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:NNTF \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 {cpsp}{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 {smpl}{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 {supr}{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:NnnTF \l_@@_fontface_cs_tl {#1} {#2}
18       {
19         \@@_update_featstr:n {\l_fontspec_feature_string_tl}
20       }
21     }
22     {
23       \@@_warning:nx {aat-feature-not-exist-in-font} {#1,#2}
24     }
25 }
```

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

`\@@_make_AAT_feature_string:NnnTF`

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 61).

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 Xe_{La}TeX 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-addfeatures} {#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   \tl_set:Nn \l_@@_font_path_tl {#1}
```

```

29     \@@_font_is_file:
30 <*XE>
31     \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
32 </XE>
33 }
34 \aliasfontfeature{Path}{ExternalLocation}
35 \@@_keys_define_code:nmn {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.

```

36 \@@_keys_define_code:nmn {fontspec-prepare-external} {Extension}
37 {
38     \tl_set:Nn \l_@@_extension_tl {#1}
39     \bool_if:NF \l_@@_external_bool
40     {
41         \keys_set:nn {fontspec-prepare-external} {Path}
42     }
43 }
44 \tl_clear:N \l_@@_extension_tl
45 \@@_keys_define_code:nmn {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.

```

46 \@@_keys_define_code:nmn {fontspec-prepare-external} {KpseOnly}
47 {
48     \bool_set_true:N \l_@@_external_kpse_bool
49     \bool_if:NT \l_@@_external_bool \@@_font_is_file:
50 }
51 \@@_keys_define_code:nmn {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.

```

52 <*XE>
53 \keys_define:nn {fontspec-renderer}
54 {
55     Renderer .choices:nn =
56     {AAT,ICU,OpenType,Graphite,Full,Basic,Node,Base,HarfBuzz,Harfbuzz}
57     {
58         \int_compare:nTF {\l_keys_choice_int <= 4}
59         {
60             \tl_set:Nx \l_@@_renderer_tl
61             {
62                 \int_case:nn \l_keys_choice_int { 1 {/AAT} 2 {/OT} 3 {/OT} 4 {/GR} }
63             }
64 <debug>\typeout{Renderer:~ \l_@@_renderer_tl}
65             \tl_gset:Nx \g_@@_single_feat_tl { \l_@@_renderer_tl }
66         }

```

```

67     {
68     \l_keys_define:nn {only-luatex-feature} {Renderer=Full/Basic/Node/Base/HarfBuzz}
69     }
70   }
71 }
72 </XE>
73 <*LU>
74 \keys_define:nn {fontspec-renderer}
75 {
76   Renderer .choices:nn =
77     {Full,Node,Basic,Base,HarfBuzz,Harfbuzz,OpenType,AAT,Graphite}
78     {
79     \int_compare:nT {\l_keys_choice_int >= 5} { \bool_set_true:N \l_@@_harfbuzz_bool }
80
81     \tl_set:Nx \l_@@_mode_tl
82     {
83     \int_case:nn \l_keys_choice_int { 1 {node} 2 {node} 3 {base} 4 {base} 5 {harf} 6 {
84     }
85
86     \tl_set:Nx \l_@@_shaper_tl
87     {
88     \int_case:nn \l_keys_choice_int { 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {ot} 8 {coretext}
89     }
90
91     <debug>\typeout{Mode:~"\l_@@_mode_tl"~/~Shaper:~"\l_@@_shaper_tl"}
92
93     \tl_gset:Nx \g_@@_single_feat_tl
94     {
95     mode=\l_@@_mode_tl ;
96     \tl_if_empty:NF \l_@@_shaper_tl { shaper=\l_@@_shaper_tl}
97     }
98   } ,
99
100   Renderer unknown .code:n =
101   {
102     \bool_set_true:N \l_@@_harfbuzz_bool
103     \l_keys_define:nn {unknown-renderer} {#1}
104     \tl_set:Nn \l_@@_mode_tl {harf}
105     \tl_set:Nn \l_@@_shaper_tl {#1}
106   } ,
107 }
108 </LU>

```

1.2 Pre-parsed features

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

```

109 \l_keys_define_code:nnn {fontspec-prepare} {Script}
110 {
111 <XE> \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
112 \tl_set:Nn \l_@@_script_name_tl {#1}

```

```
113 }
```

Exactly the same:

```
114 \@@_keys_define_code:nmn {fontspec-prepare} {Language}
115 {
116 <XE> \keys_set:n {fontspec-renderer} {Renderer=OpenType}
117 \tl_set:Nn \l_@@_lang_name_tl {#1}
118 }
```

TTC font index

```
119 \@@_keys_define_code:nmn {fontspec-prepare} {FontIndex}
120 {
121 \str_if_eq:eeF { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
122 { \@@_warning:n {font-index-needs-ttc} }
123 <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
124 <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
125 }
126 \@@_keys_define_code:nmn {fontspec} {FontIndex}
127 {
128 <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
129 <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
130 }
```

1.3 Font faces

Upright

```
131 \@@_keys_define_code:nmn {fontspec-prepare-external} {UprightFont}
132 {
133 \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
134 }
```

Italic and slanted

```
135 \@@_keys_define_code:nmn {fontspec-prepare-external} {ItalicFont}
136 {
137 \tl_if_empty:nTF {#1}
138 {
139 \bool_set_true:N \l_@@_noit_bool
140 }
141 {
142 \bool_set_false:N \l_@@_noit_bool
143 \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
144 }
145 }
146 \@@_keys_define_code:nmn {fontspec-prepare-external} {SlantedFont}
147 {
148 \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
149 }
```



```

150 \@@_keys_define_code:nmn {fontspec-prepare-external} {SwashFont}
151 {
152   \fontspec_complete_fontname:Nn \l_@@_fontname_sw_tl {#1}
153 }

```

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!

```

154 %\@@_keys_define_code:nmn {fontspec-prepare-external} {BoldSeries}
155 % {
156 %   \tl_gset:Nx \g_@@_curr_series_tl { #1 }
157 %   \seq_put_right:Nx \l_@@_bf_series_seq { #1 }
158 % }

```

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

```

159 \@@_keys_define_code:nmn {fontspec-prepare-external} {BoldFont}
160 {
161   \tl_if_empty:nTF {#1}
162   {
163     \bool_set_true:N \l_@@_nobf_bool
164   }
165   {
166     \bool_set_false:N \l_@@_nobf_bool
167     \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
168
169     \seq_if_empty:NT \l_@@_bf_series_seq
170     {
171       \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
172       \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
173     }
174
175     \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
176     {
177       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
178     }
179
180     \prop_put:NxV \l_@@_nfss_prop {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
181
182     <debug>\typeout{Setting-bold-font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
183
184   }
185 }

```

Bold italic/slanted

```

186 \@@_keys_define_code:nmn {fontspec-prepare-external} {BoldItalicFont}
187 {
188   \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
189 }
190 \@@_keys_define_code:nmn {fontspec-prepare-external} {BoldSlantedFont}
191 {

```

```

192     \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
193   }
194 \@@_keys_define_code:nmn {fontspec-preparse-external} {BoldSwashFont}
195   {
196     \fontspec_complete_fontname:Nn \l_@@_fontname_bfsw_tl {#1}
197   }

```

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

```

198 \@@_keys_define_code:nmn {fontspec} {SmallCapsFont}
199   {
200     \tl_if_empty:nTF {#1}
201       {
202         \bool_set_true:N \l_@@_nosc_bool
203       }
204       {
205         \bool_set_false:N \l_@@_nosc_bool
206         \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
207       }
208   }

```

1.3.1 Prepared font features

```

209 \@@_keys_define_code:nmn {fontspec-preparse} {UprightFeatures}
210   {
211     \clist_put_right:Nn \l_@@_fontfeat_up_clist {#1}
212   }
213 \@@_keys_define_code:nmn {fontspec-preparse} {BoldFeatures}
214   {
215     \clist_put_right:Nn \l_@@_fontfeat_bf_clist {#1}
216     % \prop_put:NxV \l_@@_nfss_prop
217     %   {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
218   }
219 \@@_keys_define_code:nmn {fontspec-preparse} {ItalicFeatures}
220   {
221     \clist_put_right:Nn \l_@@_fontfeat_it_clist {#1}
222   }
223 \@@_keys_define_code:nmn {fontspec-preparse} {BoldItalicFeatures}
224   {
225     \clist_put_right:Nn \l_@@_fontfeat_bfit_clist {#1}
226   }
227 \@@_keys_define_code:nmn {fontspec-preparse} {SlantedFeatures}
228   {
229     \clist_put_right:Nn \l_@@_fontfeat_sl_clist {#1}
230   }
231 \@@_keys_define_code:nmn {fontspec-preparse} {BoldSlantedFeatures}
232   {
233     \clist_put_right:Nn \l_@@_fontfeat_bfsl_clist {#1}
234   }
235 \@@_keys_define_code:nmn {fontspec-preparse} {SwashFeatures}
236   {

```

```

237     \clist_put_right:Nn \l_@@_fontfeat_sw_clist {#1}
238   }
239   \@@_keys_define_code:nmn {fontspec-prepare} {BoldSwashFeatures}
240   {
241     \clist_put_right:Nn \l_@@_fontfeat_bfsw_clist {#1}
242   }

```

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

```

243   \@@_keys_define_code:nmn {fontspec} {SmallCapsFeatures}
244   {
245     \bool_if:NF \l_@@_firsttime_bool
246     {
247       \clist_put_right:Nn \l_@@_fontfeat_sc_clist {#1}
248     }
249   }

```

Features varying by size

```

250   \@@_keys_define_code:nmn {fontspec-prepare} {SizeFeatures}
251   {
252     \clist_set:Nn \l_@@_sizefeat_clist {#1}
253     \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
254   }
255   \@@_keys_define_code:nmn {fontspec-prepare-nested} {SizeFeatures}
256   {
257     \clist_set:Nn \l_@@_sizefeat_clist {#1}
258     \tl_if_empty:NT \l_@@_this_font_tl
259     { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
260   }
261   \@@_keys_define_code:nmn {fontspec-prepare-nested} {Font}
262   {
263     \tl_set:Nn \l_@@_this_font_tl {#1}
264   }
265   \@@_keys_define_code:nmn {fontspec} {SizeFeatures}
266   {
267     % dummy
268   }
269   \@@_keys_define_code:nmn {fontspec} {Font}
270   {
271     % dummy
272   }
273   \@@_keys_define_code:nmn {fontspec-sizing} {Size}
274   {
275     \tl_set:Nn \l_@@_size_tl {#1}
276   }
277   \@@_keys_define_code:nmn {fontspec-sizing} {Font}
278   {
279     \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
280   }

```

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

```

281   \@@_keys_define_code:nmn {fontspec-opentype} {UprightFont} {}
282   \@@_keys_define_code:nmn {fontspec-opentype} {ItalicFont} {}

```

```

283 \@@_keys_define_code:nmn {fontspec-opentype} {SlantedFont} {}
284 \@@_keys_define_code:nmn {fontspec-opentype} {BoldFont} {}
285 \@@_keys_define_code:nmn {fontspec-opentype} {BoldItalicFont} {}
286 \@@_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.

```

287 \@@_keys_define_code:nmn {fontspec-preparse} {NFSSEncoding}
288 {
289   \tl_gset:Nx \g_@@_nfss_enc_tl { #1 }
290 }

```

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.)

```

291 \@@_keys_define_code:nmn {fontspec-preparse} {NFSSFamilY}
292 {
293   \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
294 }

```

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

```

295 \@@_keys_define_code:nmn {fontspec-preparse} {FontFace}
296 {
297   \tl_clear:N \l_@@_this_font_tl
298   \clist_set:No \l_@@_arg_clist { \use_iii:nmn #1 }
299   \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
300   \int_compare:nT { \clist_count:N \l_@@_arg_clist = 1 }
301   {
302     <debug>\typeout{FontFace~ parsing:~ one~ clist~ item}
303     \tl_if_in:NnF \l_@@_arg_clist {=}
304     {
305       <debug>\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
306       \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
307       \tl_clear:N \l_@@_this_feat_clist
308     }
309   }
310
311   \@@_add_nfssfont:nmnn
312   { \use_i:nmn #1 } { \use_ii:nmn #1 } { \l_@@_this_font_tl } { \l_@@_this_feat_clist }
313 }

```

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:nm` do all the work in the auto-scaling cases.

```

314 \@@_keys_define_code:nmn {fontspec} {Scale}
315 {

```

```

316 \str_case:nnF {#1}
317 {
318   {MatchLowercase} { \@@_calc_scale:n {5} }
319   {MatchUppercase} { \@@_calc_scale:n {8} }
320   {MatchAveragecase} { \@@_calc_scale:nn {5} {8} }
321 }
322 { \tl_set:Nx \l_@@_scale_tl {#1} }
323 \@@_info:n {set-scale}
324 }

```

ScaleAgain

```

325 \@@_keys_define_code:nnm {fontspec} {ScaleAgain}
326 {
327   \tl_if_empty:NT \l_@@_scale_tl { \tl_set:Nn \l_@@_scale_tl {1} }
328   \tl_set:Nx \l_@@_scale_tl { \fp_eval:n { #1 * \l_@@_scale_tl } }
329   \@@_info:n {set-scale}
330 }

```

`\@@_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 `csrmfamily` has been overwritten. (Note that changing `\rmfamily` with `fontspec` resets `\encodingdefault` appropriately.)

```

331 \cs_new:Nn \@@_calc_scale:n
332 {
333   \group_begin:
334
335   \fontencoding { \encodingdefault }
336   \fontfamily { \familydefault }
337   \selectfont
338
339   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
340   \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_@@_fontface_cs_tl
341
342   \tl_set:Nx \l_@@_scale_tl
343   {
344     \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
345                 \dim_to_fp:n {\l_@@_tmpb_dim} }
346   }
347
348   \exp_args:NNNx
349   \group_end:
350   \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
351 }

```

(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.

```

352 \cs_new:Nn \@@_calc_scale:nn
353 {
354   \group_begin:
355     \__fontspec_calc_scale:n {#1}
356     \tl_set_eq:NN \l_@@_tmp_tl \l_@@_scale_tl
357     \__fontspec_calc_scale:n {#2}
358     \tl_set:Nx \l_@@_scale_tl
359       {
360         \fp_eval:n { (\l_@@_tmp_tl + \l_@@_scale_tl)/2 }
361       }
362     \exp_args:NNNx
363     \group_end:
364     \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
365 }

```

(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’.

```

366 \cs_new:Nn \@@_set_font_dimen:NnN
367 {
368   \dim_set:Nn #1 { \fontdimen #2 #3 }
369   \dim_compare:nNnT #1 = {0pt}
370     {
371       \settoheight #1
372         {
373           \str_if_eq:nnTF {#3} {\font} \rmfamily #3
374           \int_case:nnF #2
375             {
376               {5} {x} % x-height
377               {8} {X} % cap-height
378             } {?} % "else" clause; never reached.
379         }
380     }
381 }

```

(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.

```

382 \@@_keys_define_code:nmn {fontspec} {WordSpace}
383 {
384   \bool_if:NF \l_@@_firsttime_bool
385     { \_fontspec_parse_wordspace:w #1,,,\q_stop }
386 }
387 \@@_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}`.

```

388 \cs_set:Npn \fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
389   {
390     \tl_if_empty:nTF {#4}
391     {
392       \tl_set:Nn \l_@@_wordspace_adjust_tl
393       {
394         \fontdimen 2 \font = #1 \fontdimen 2 \font
395         \fontdimen 3 \font = #1 \fontdimen 3 \font
396         \fontdimen 4 \font = #1 \fontdimen 4 \font
397       }
398     }
399     {
400       \tl_set:Nn \l_@@_wordspace_adjust_tl
401       {
402         \fontdimen 2 \font = #1 \fontdimen 2 \font
403         \fontdimen 3 \font = #2 \fontdimen 3 \font
404         \fontdimen 4 \font = #3 \fontdimen 4 \font
405       }
406     }
407   }

```

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

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

```

408 \@@_keys_define_code:nmm {fontspec} {PunctuationSpace}
409   {
410     \str_case_e:nnF {#1}
411     {
412       {WordSpace}
413       {
414         \tl_set:Nn \l_@@_punctspace_adjust_tl
415         { \fontdimen 7 \font = 0 \fontdimen 2 \font }
416       }
417       {TwiceWordSpace}
418       {
419         \tl_set:Nn \l_@@_punctspace_adjust_tl
420         { \fontdimen 7 \font = 1 \fontdimen 2 \font }
421       }
422     }
423     {
424       \tl_set:Nn \l_@@_punctspace_adjust_tl
425       { \fontdimen 7 \font = #1 \fontdimen 7 \font }
426     }
427   }
428 \@@_aff_error:n {PunctuationSpace}

```

Secret hook into the font-adjustment code

```

429 \@@_keys_define_code:nmm {fontspec} {FontAdjustment}

```

```

430 {
431   \tl_put_right:Nx \l_@@_postadjust_tl {#1}
432 }

```

Letterspacing

```

433 \@@_keys_define_code:nmn {fontspec} {LetterSpace}
434 {
435   \@@_update_featstr:n {letterspace=#1}
436 }

```

Hyphenation character This feature takes one of three arguments: ‘None’, $\langle glyph \rangle$, or $\langle slot \rangle$. 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.

```

437 \@@_keys_define_code:nmn {fontspec} {HyphenChar}
438 {
439   \str_if_eq:nnTF {#1} {None}
440   {
441     \tl_put_right:Nn \l_@@_postadjust_tl
442       { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
443   }
444   {
445     \LU \@@_warning:nx {only-xetex-feature} {HyphenChar}
446
447     \tl_if_single:nTF {#1}
448       { \tl_set:Nn \l_@@_hyphenchar_tl {`#1} }
449       { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }
450
451     \exp_args:No \@@_primitive_font_glyph_if_exist:NnTF \l_@@_fontface_cs_tl {\l_@@_hyphen
452       {
453         \tl_put_right:Nn \l_@@_postadjust_tl
454           { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
455       }
456       { \@@_error:nxx {no-glyph}{\l_fontspeg_fontname_tl}{#1} }
457
458     }
459   }
460 \@@_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 $\backslash color\langle name \rangle$. If this fails the argument is assumed to be a hex color.

```

461 \@@_keys_define_code:nmn {fontspec} {Color}
462 {
463   \*XE
464   \color_if_exist:nTF {#1}
465   {
466     \color_export:nnN {#1} {HTML}\l_@@_hexcol_tl
467   }

```



```

468     {
469       \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
470       {
471         \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
472       }
473       {
474         \int_compare:nTF { \tl_count:n {#1} == 6 }
475         { \tl_set:Nn \l_@@_hexcol_tl {#1} }
476         {
477           \int_compare:nTF { \tl_count:n {#1} == 8 }
478           { \fontspec_parse_colour:viii #1 }
479           {
480             \bool_if:NF \l_@@_firsttime_bool
481             { \@@_warning:nx {bad-colour} {#1} }
482           }
483         }
484       }
485     }
486 \
```

```

519     }
520     \tl_set:Nn \l_@@_opacity_tl {#7#8}
521   }
522 \aliasfontfeature{Color}{Colour}
523 \@@_keys_define_code:nmn {fontspec} {Opacity}
524 {
525   \int_set:Nn \l_@@_tmp_int {255}
526   \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
527   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
528     {
529       \bool_if:NF \l_@@_firsttime_bool
530       { \@@_warning:nx {opa-twice} {#1} }
531     }
532   \tl_set:Nx \l_@@_opacity_tl
533     {
534 <LU>     ,
535           \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
536           \int_to_hex:n { \l_@@_tmp_int }
537     }
538   }

```

Mapping

```

539 <*XE>
540 \@@_keys_define_code:nmn {fontspec-aat} {Mapping}
541 {
542   \tl_set:Nn \l_@@_mapping_tl { #1 }
543 }
544 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
545 {
546   \tl_set:Nn \l_@@_mapping_tl { #1 }
547 }
548 </XE>
549 <*LU>
550 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
551 {
552   \str_if_eq:nnTF {#1} {tex-text}
553     {
554       \@@_warning:n {no-mapping-ligtex}
555       \msg_redirect_name:nmn {fontspec} {no-mapping-ligtex} {none}
556       \keys_set:nm {fontspec-opentype} { Ligatures=TeX }
557     }
558   { \@@_warning:n {no-mapping} }
559 }
560 </LU>

```

1.4.1 Continuous font axes

```

561 <*XE>
562 \@@_keys_define_code:nmn {fontspec} {Weight}
563 {

```

```

564     \@@_update_featstr:n{weight=#1}
565   }
566 </XE>
567 <LU>\@@_define_opentype_variation_axis:nn {Weight} {wght}
568 <*XE>
569 \@@_keys_define_code:nmn {fontspec} {Width}
570   {
571     \@@_update_featstr:n{width=#1}
572   }
573 </XE>
574 <LU>\@@_define_opentype_variation_axis:nn {Width} {wdth}
575 \@@_define_opentype_variation_axis:nn {Slant} {slnt}
576 \@@_keys_define_code:nmn {fontspec} {OpticalSize}
577 <*XE>
578   {
579     \bool_if:NTF \l_@@_ot_bool
580     {
581       \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
582     }
583     {
584       \bool_if:NT \l_@@_mm_bool
585       {
586         \@@_update_featstr:n { optical size = #1 }
587       }
588     }
589     \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
590     {
591       \bool_if:NT \l_@@_firsttime_bool
592       { \@@_warning:nx {no-opticals} {\l_fontspec_fontname_tl} }
593     }
594   }
595 </XE>
596 <*LU>
597   {
598     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
599   }
600 </LU>

```

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

```

601 \@@_keys_define_code:nmn {fontspec-opentype} {RawAxis}
602   {
603     \prop_gput_from_keyval:Nn \g_@@_rawvariations_prop {#1}
604   }

```

1.4.2 Variation instances

```

605 \@@_keys_define_code:nmn {fontspec-opentype} {Instance}
606   {
607     \tl_gset:Nn \g_@@_instance_tl {#1}
608   }

```

1.4.3 Font transformations

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

```

609 \keys_define:nn {fontspec}
610 {
611   FakeSlant .code:n =
612     {
613       \@@_update_featstr:n {slant=#1}
614     },
615   FakeSlant .default:n = {0.2}
616 }
617 \keys_define:nn {fontspec}
618 {
619   FakeStretch .code:n =
620     {
621       \@@_update_featstr:n {extend=#1}
622     },
623   FakeStretch .default:n = {1.2}
624 }
625 \keys_define:nn {fontspec}
626 {
627   FakeBold .code:n =
628     {
629       \@@_update_featstr:n {embolden=#1}
630     },
631   FakeBold .default:n = {1.5}
632 }

```

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`.)

```

633 \keys_define:nn {fontspec}
634 {
635   AutoFakeSlant .code:n =
636     {
637       \bool_if:NT \l_@@_firsttime_bool
638       {
639         \tl_set:Nn \l_@@_fake_slant_tl {#1}
640         \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
641         \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
642         \bool_set_false:N \l_@@_noit_bool
643
644         \tl_if_empty:NF \l_@@_fake_embolden_tl
645         {
646           \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
647             {FakeBold=\l_@@_fake_embolden_tl}
648           \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
649           \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
650         }
651       }

```

```

652     },
653     AutoFakeSlant .default:n = {0.2}
654 }

```

Same but reversed:

```

655 \keys_define:nm {fontspec}
656 {
657     AutoFakeBold .code:n =
658     {
659         \bool_if:NT \l_@@_firsttime_bool
660         {
661             \tl_set:Nn \l_@@_fake_embolden_tl {#1}
662             \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
663             \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
664             \bool_set_false:N \l_@@_nobf_bool
665
666             \tl_if_empty:NF \l_@@_fake_slant_tl
667             {
668                 \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
669                 {FakeSlant=\l_@@_fake_slant_tl}
670                 \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
671                 \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
672             }
673         }
674     },
675     AutoFakeBold .default:n = {1.5}
676 }

```

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.

```

677 \@@_keys_define_code:nmm {fontspec-opentype} {RawFeature}
678 {
679     \@@_update_featstr:n {#1}
680 }
681 \@@_keys_define_code:nmm {fontspec-aat} {RawFeature}
682 {
683     \@@_update_featstr:n {#1}
684 }

```

File XIV

fontspec-code-feat-opentype.dtx

1 OpenType feature definitions

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

2 Regular key=val / tag definitions

2.1 Ligatures

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

Emulate CM extra ligatures.

```
19 \langle *XE \rangle
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 \langle /XE \rangle
27 \langle LU \rangle \@@_define_opentype_onoffreset:nmmmm {Ligatures} {TeX} {} {tlig} {}
```

2.2 Letters

```
28 \@@_define_opentype_feature_group:n {Letters}
29 \@@_define_opentype_feature:nmmmm {Letters} {ResetAll} {} {}
30 {
31  \langle LU \rangle +lower,-lower,+upper,-upper,+case,+cpsp,
32   +smcp,+pcap,+c2sc,+c2pc,+unic,+rand,
33   -smcp,-pcap,-c2sc,-c2pc,-unic,-rand
34  }
35 \langle *LU \rangle
```

```

36 \keys_define:nn {fontspec-opentype}
37 {
38   Letters / Uppercase .code:n = {
39     \@@_make_OT_feature:nnn {} {+upper} {+lower}
40     \@@_make_OT_feature:nnn {} {+case} {}
41     \@@_make_OT_feature:nnn {} {+csp} {}
42   },
43 }
44 \@@_define_opentype_feature:nnnnn {Letters} {UppercaseOff} {} {-upper} {+case,+csp}
45 \@@_define_opentype_feature:nnnnn {Letters} {UppercaseReset} {} {} {+upper,-upper}
46 \@@_define_opentype_onoffreset:nnnnn {Letters} {Lowercase} {} {lower} {+upper,+case,+csp}
47 /LU
48 \@@_define_opentype_onoffreset:nnnnn {Letters} {SmallCaps} {smcp} {smcp} {+pcap,+unic}
49 \@@_define_opentype_onoffreset:nnnnn {Letters} {PetiteCaps} {pcap} {pcap} {+smcp,+unic}
50 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercaseSmallCaps} {c2sc} {c2sc} {+c2pc,+unic}
51 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercasePetiteCaps} {c2pc} {c2pc} {+c2sc,+unic}
52 \@@_define_opentype_onoffreset:nnnnn {Letters} {Unicase} {unic} {unic} {}
53 \@@_define_opentype_onoffreset:nnnnn {Letters} {Random} {rand} {rand} {}

```

2.3 Numbers

```

54 \@@_define_opentype_feature_group:n {Numbers}
55 \@@_define_opentype_feature:nnnnn {Numbers} {ResetAll} {} {}
56 {
57   +tnum,-tnum,
58   +pnum,-pnum,
59   +onum,-onum,
60   +lnum,-lnum,
61   +zero,-zero,
62   +anum,-anum,
63 }
64 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Monospaced} {tnum} {tnum} {+pnum,-pnum}
65 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Proportional} {pnum} {pnum} {+tnum,-tnum}
66 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Lowercase} {onum} {onum} {+lnum,-lnum}
67 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Uppercase} {lnum} {lnum} {+onum,-onum}
68 \@@_define_opentype_onoffreset:nnnnn {Numbers} {SlashedZero} {zero} {zero} {}
69 \aliasfontfeatureoption {Numbers} {Monospaced} {Tabular}
70 \aliasfontfeatureoption {Numbers} {Lowercase} {OldStyle}
71 \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.

```

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

```

2.4 Vertical position

```

73 \@@_define_opentype_feature_group:n {VerticalPosition}
74 \@@_define_opentype_feature:nnnnn {VerticalPosition} {ResetAll} {} {}
75 {
76   +sups,-sups,
77   +subs,-subs,
78   +ordn,-ordn,

```

```

79     +numr,-numr,
80     +dnom,-dnom,
81     +sinf,-sinf,
82   }
83 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Superior}           {supr} {supr} {+s}
84 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Inferior}           {subr} {subr} {+s}
85 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Ordinal}             {ordn} {ordn} {+s}
86 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Numerator}           {numr} {numr} {+s}
87 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Denominator}         {dnom} {dnom} {+s}
88 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {ScientificInferior} {sinf} {sinf} {+s}

```

2.5 Contextuals

```

89 \@@_define_opentype_feature_group:n {Contextuals}
90 \@@_define_opentype_feature:nnnnn {Contextuals} {ResetAll} {} {}
91   {
92     +cswh,-cswh,
93     +calt,-calt,
94     +init,-init,
95     +fina,-fina,
96     +falt,-falt,
97     +medi,-medi,
98   }
99 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Swash}           {cswh} {cswh} {}
100 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Alternate}       {calt} {calt} {}
101 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordInitial}     {init} {init} {}
102 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordFinal}       {fina} {fina} {}
103 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {LineFinal}       {falt} {falt} {}
104 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Inner}           {medi} {medi} {}

```

2.6 Diacritics

```

105 \@@_define_opentype_feature_group:n {Diacritics}
106 \@@_define_opentype_feature:nnnnn {Diacritics} {ResetAll} {} {}
107   {
108     +mark,-mark,
109     +mkmk,-mkmk,
110     +abvm,-abvm,
111     +blwm,-blwm,
112   }
113 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToBase} {mark} {mark} {}
114 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToMark} {mkmk} {mkmk} {}
115 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {AboveBase} {abvm} {abvm} {}
116 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {BelowBase} {blwm} {blwm} {}

```

2.7 Kerning

```

117 \@@_define_opentype_feature_group:n {Kerning}
118 \@@_define_opentype_feature:nnnnn {Kerning} {ResetAll} {} {}
119   {
120     +cspc,-cspc,
121     +kern,-kern,
122   }

```



```

123 \@@_define_opentype_onoffreset:nmnnn {Kerning} {Uppercase} {csp} {csp} {}
124 \@@_define_opentype_feature:nmnnn {Kerning} {On} {kern} {+kern} {-kern}
125 \@@_define_opentype_feature:nmnnn {Kerning} {Off} {kern} {-kern} {+kern}
126 \@@_define_opentype_feature:nmnnn {Kerning} {Reset} {} {} {+kern,-kern}

```

2.8 Fractions

```

127 \@@_define_opentype_feature_group:n {Fractions}
128 \@@_define_opentype_feature:nmnnn {Fractions} {ResetAll} {} {}
129 {
130   +frac,-frac,
131   +afrc,-afrc,
132 }
133 \@@_define_opentype_feature:nmnnn {Fractions} {On} {frac} {+frac} {}
134 \@@_define_opentype_feature:nmnnn {Fractions} {Off} {frac} {-frac} {}
135 \@@_define_opentype_feature:nmnnn {Fractions} {Reset} {} {} {+frac,-frac}
136 \@@_define_opentype_onoffreset:nmnnn {Fractions} {Alternate} {afrc} {afrc} {-frac}
137 \@@_define_opentype_feature_group:n {LocalForms}
138 \@@_define_opentype_feature:nmnnn {LocalForms} {On} {locl} {+locl} {}
139 \@@_define_opentype_feature:nmnnn {LocalForms} {Off} {locl} {-locl} {}
140 \@@_define_opentype_feature:nmnnn {LocalForms} {Reset} {} {} {+locl,-locl}

```

2.9 Style

```

141 \@@_define_opentype_feature_group:n {Style}
142 \@@_define_opentype_feature:nmnnn {Style} {ResetAll} {} {}
143 {
144   +salt,-salt,
145   +ital,-ital,
146   +ruby,-ruby,
147   +swsh,-swsh,
148   +hist,-hist,
149   +titl,-titl,
150   +hkna,-hkna,
151   +vkna,-vkna,
152   +ssty=0,-ssty=0,
153   +ssty=1,-ssty=1,
154 }
155 \@@_define_opentype_onoffreset:nmnnn {Style} {Alternate} {salt} {salt} {}
156 \@@_define_opentype_onoffreset:nmnnn {Style} {Italic} {ital} {ital} {}
157 \@@_define_opentype_onoffreset:nmnnn {Style} {Ruby} {ruby} {ruby} {}
158 \@@_define_opentype_onoffreset:nmnnn {Style} {Swash} {swsh} {swsh} {}
159 \@@_define_opentype_onoffreset:nmnnn {Style} {Cursive} {swsh} {curs} {}
160 \@@_define_opentype_onoffreset:nmnnn {Style} {Historic} {hist} {hist} {}
161 \@@_define_opentype_onoffreset:nmnnn {Style} {Titling} {titl} {titl} {}
162 \@@_define_opentype_onoffreset:nmnnn {Style} {TitlingCaps} {titl} {titl} {} % backwards
163 \@@_define_opentype_onoffreset:nmnnn {Style} {HorizontalKana} {hkna} {hkna} {+vkna,+pkna}
164 \@@_define_opentype_onoffreset:nmnnn {Style} {VerticalKana} {vkna} {vkna} {+hkna,+pkna}
165 \@@_define_opentype_onoffreset:nmnnn {Style} {ProportionalKana} {pkna} {pkna} {+vkna,+hkna}
166 \@@_define_opentype_feature:nmnnn {Style} {MathScript} {ssty} {+ssty=0} {+ssty=1}
167 \@@_define_opentype_feature:nmnnn {Style} {MathScriptScript} {ssty} {+ssty=1} {+ssty=0}
168 \@@_define_opentype_onoffreset:nmnnn {Style} {Uppercase} {case} {case} {}

```

2.10 CJK shape

```
169 \@@_define_opentype_feature_group:n {CJKShape}
170 \@@_define_opentype_feature:n:n:n {CJKShape} {ResetAll} {} {}
171 {
172   +trad,-trad,
173   +smp1,-smp1,
174   +jp78,-jp78,
175   +jp83,-jp83,
176   +jp90,-jp90,
177   +jp04,-jp04,
178   +expt,-expt,
179   +nlck,-nlck,
180 }
181 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {Traditional} {trad} {trad} {+smp1,+jp78,+jp83}
182 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {Simplified} {smp1} {smp1} {+trad,+jp78,+jp83}
183 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {JIS1978} {jp78} {jp78} {+trad,+smp1,+jp83}
184 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {JIS1983} {jp83} {jp83} {+trad,+smp1,+jp78}
185 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {JIS1990} {jp90} {jp90} {+trad,+smp1,+jp78}
186 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {JIS2004} {jp04} {jp04} {+trad,+smp1,+jp78}
187 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {Expert} {expt} {expt} {+trad,+smp1,+jp78}
188 \@@_define_opentype_onoffreset:n:n:n {CJKShape} {NLC} {nlck} {nlck} {+trad,+smp1,+jp78}
```

2.11 Character width

```
189 \@@_define_opentype_feature_group:n {CharacterWidth}
190 \@@_define_opentype_feature:n:n:n {CharacterWidth} {ResetAll} {} {}
191 {
192   +pwid,-pwid,
193   +fwid,-fwid,
194   +hwid,-hwid,
195   +twid,-twid,
196   +qwid,-qwid,
197   +palt,-palt,
198   +halt,-halt,
199 }
200 \@@_define_opentype_onoffreset:n:n:n {CharacterWidth} {Proportional} {pwid} {pwid} {+}
201 \@@_define_opentype_onoffreset:n:n:n {CharacterWidth} {Full} {fwid} {fwid} {+}
202 \@@_define_opentype_onoffreset:n:n:n {CharacterWidth} {Half} {hwid} {hwid} {+}
203 \@@_define_opentype_onoffreset:n:n:n {CharacterWidth} {Third} {twid} {twid} {+}
204 \@@_define_opentype_onoffreset:n:n:n {CharacterWidth} {Quarter} {qwid} {qwid} {+}
205 \@@_define_opentype_onoffreset:n:n:n {CharacterWidth} {AlternateProportional} {palt} {palt} {+}
206 \@@_define_opentype_onoffreset:n:n:n {CharacterWidth} {AlternateHalf} {halt} {halt} {+}
```

2.12 Vertical

According to spec vkrn must also activate vp1 if available but for simplicity we don't do that here (yet?).

```
207 \@@_define_opentype_feature_group:n {Vertical}
208 \@@_define_opentype_onoffreset:n:n:n {Vertical} {RotatedGlyphs} {vrt2} {vrt2} {+vrtr,+}
209 \@@_define_opentype_onoffreset:n:n:n {Vertical} {AlternatesForRotation} {vrtr} {vrtr} {+vrt2}
210 \@@_define_opentype_onoffreset:n:n:n {Vertical} {Alternates} {vert} {vert} {+vrt2}
```

```

211 \@@_define_opentype_onoffreset:nnnnn {Vertical} {KanaAlternates}      {vkna} {vkna} {+hkna}
212 \@@_define_opentype_onoffreset:nnnnn {Vertical} {Kerning}           {vkern} {vkern} {}
213 \@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternateMetrics}  {valt} {valt} {+vhal,+
214 \@@_define_opentype_onoffreset:nnnnn {Vertical} {HalfMetrics}      {vhal} {vhal} {+valt,+
215 \@@_define_opentype_onoffreset:nnnnn {Vertical} {ProportionalMetrics} {vpal} {vpal} {+valt,+

```

3 OpenType features that need numbering

3.1 Alternate

```

216 \@@_define_opentype_feature_group:n {Alternate}
217 \keys_define:nn {fontspec-opentype}
218 {
219   Alternate .default:n = {0} ,
220   Alternate .groups:n = {opentype},
221   Alternate / unknown .code:n =
222     {
223       \clist_map_inline:nn {#1}
224         { \@@_make_OT_feature:nnn {salt}{ +salt = ##1 }{} }
225     }
226 }
227 <*LU>
228 \keys_define:nn {fontspec-opentype}
229 {
230   Alternate / Random .code:n =
231     { \@@_make_OT_feature:nnn {salt}{ +salt = random }{} } ,
232 }
233 </LU>
234 \aliasfontfeature{Alternate}{StylisticAlternates}

```

3.2 Variant / StylisticSet

```

235 \@@_define_opentype_feature_group:n {Variant}
236 \keys_define:nn {fontspec-opentype}
237 {
238   Variant .default:n = {0} ,
239   Variant .groups:n = {opentype} ,
240   Variant / unknown .code:n =
241     {
242       \clist_map_inline:nn {#1}
243         {
244           \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
245         }
246     }
247 }
248 \aliasfontfeature{Variant}{StylisticSet}

```

3.3 CharacterVariant

```

249 \@@_define_opentype_feature_group:n {CharacterVariant}
250 \use:x

```

```

251 {
252   \cs_new:Npn \exp_not:N \fontspec_parse_cv:w
253     ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
254     {
255       \@@_make_OT_feature:xxx
256       { cv \exp_not:N \two@digits {##1} }
257       { +cv \exp_not:N \two@digits {##1} = ##2 } {}
258     }
259   \keys_define:nn {fontspec-opentype}
260     {
261       CharacterVariant / unknown .code:n =
262       {
263         \clist_map_inline:nn {##1}
264         {
265           \exp_not:N \fontspec_parse_cv:w
266           #####1 \c_colon_str 0 \c_colon_str \exp_not:N \q_nil
267         }
268       }
269     }
270 }

```

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

3.4 Annotation

```

271 \@@_define_opentype_feature_group:n {Annotation}
272 \keys_define:nn {fontspec-opentype}
273 {
274   Annotation .default:n = {0} ,
275   Annotation .groups:n = {opentype},
276   Annotation / unknown .code:n =
277   {
278     \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
279   }
280 }

```

3.5 Ornament

```

281 \@@_define_opentype_feature_group:n {Ornament}
282 \keys_define:nn {fontspec-opentype}
283 {
284   Ornament .default:n = {0} ,
285   Ornament .groups:n = {opentype},
286   Ornament / unknown .code:n =
287   {
288     \@@_make_OT_feature:nnn {ornm} {+ornm=#1} {}
289   }
290 }

```

4 Script and Language

4.1 Script

```

291 \keys_define:nn {fontspec-opentype}

```

```

292 {
293   Script .choice: ,
294   Script .groups:n = {opentype} ,
295 }
296 \cs_new:Nn \fontspec_new_script:nn
297 {
298   \keys_define:nn {fontspec-opentype} { Script / #1 .code:n =
299     {
300 <debug>\typeout{Trying~ [Script=#1]}
301       \bool_set_false:N \l_@@_scriptlang_exist_bool
302       \clist_map_inline:nn {#2}
303       {
304         \exp_args:No \@@_check_script:NnT \l_@@_fontface_cs_tl {####1}
305         {
306 <debug>\typeout{Script~tag~found:~####1}
307           \tl_set:Nn \l_@@_script_name_tl {#1}
308           \tl_set:Nn \l_@@_script_tl {####1}
309           \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
310           \bool_set_true:N \l_@@_scriptlang_exist_bool
311           \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
312           \clist_map_break:
313         }
314       }

```

If not found give a warning but load it anyway:

```

315   \bool_if:NF \l_@@_scriptlang_exist_bool
316   {
317 <debug>\typeout{Script~not~found!}
318     \@@_warning:nxx {no-script} {\l_fontspec_fontname_tl} {#1}
319     \clist_set:Nn \l_tmpa_clist {#2}
320     \clist_get:NN \l_tmpa_clist \l_@@_script_tl
321     \exp_args:Noo \@@_check_script:NnF \l_@@_fontface_cs_tl \l_@@_script_tl
322     {
323       \tl_set:Nn \l_@@_script_name_tl {#1}
324       \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
325       \tl_gset:Nx \g_@@_single_feat_tl { script=\l_@@_script_tl }
326     }
327   }
328 }
329 }
330 }
331 \cs_new:Nn \fontspec_default_script:nn
332 {
333   \keys_define:nn {fontspec-opentype} { Script / #1 .code:n =
334     {
335 <debug>\typeout{Trying~ [Script=#1:#2]}
336       \bool_set_false:N \l_@@_scriptlang_exist_bool
337       \clist_map_inline:nn {#2}
338       {
339         \exp_args:No \@@_check_script:NnT \l_@@_fontface_cs_tl {####1}
340         {
341 <debug>\typeout{Script~tag~found:~####1}

```

```

342         \tl_set:Nn \l_@@_script_name_tl {#1}
343         \tl_set:Nn \l_@@_script_tl {####1}
344         \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
345         \bool_set_true:N \l_@@_scriptlang_exist_bool
346         \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
347         \clist_map_break:
348     }
349 }
350 \bool_if:NF \l_@@_scriptlang_exist_bool
351 {
352 <debug>\typeout{Script-not-found!}
353     \tl_clear:N \l_@@_script_name_tl
354 }
355 }
356 }
357 }

```

When script is not explicitly requested, use this list:

```

358 \fontspec_default_script:nn {CustomDefault} {latn,DFLT}

```

4.2 Language

```

359 \keys_define:nn {fontspec-opentype}
360 {
361     Language .choice: ,
362     Language .groups:n = {opentype} ,
363 }
364 \cs_new:Nn \fontspec_new_lang:nn
365 {
366     \keys_define:nn {fontspec-opentype} { Language / #1 .code:n =
367     {
368         \bool_set_false:N \l_@@_scriptlang_exist_bool
369         \clist_map_inline:nn {#2}
370         {
371             \exp_args:No \@@_check_lang:NnTF \l_@@_fontface_cs_tl {####1}
372             {
373                 \tl_set:Nn \l_@@_lang_tl {####1}
374                 \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
375                 \tl_gset:Nx \g_@@_single_feat_tl { language=####1 }
376                 \bool_set_true:N \l_@@_scriptlang_exist_bool
377                 \clist_map_break:
378             }
379         }
380     }

```

If not found give a warning but load it anyway:

```

380         \bool_if:NF \l_@@_scriptlang_exist_bool
381         {
382 <debug>\typeout{Lang-not-found!}
383             \@@_warning:nx {language-not-exist} {#1}
384             \clist_set:Nn \l_tmpa_clist {#2}
385             \clist_get:NN \l_tmpa_clist \l_@@_lang_tl
386             \exp_args:Noo \@@_check_script:NnF \l_@@_fontface_cs_tl \l_@@_script_tl
387             {
388                 \tl_set:Nn \l_@@_script_name_tl {#1}

```

```

389         \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
390         \tl_gset:Nx \g_@@_single_feat_tl { language=###1 }
391     }
392 }
393 }
394 }
395 }

```

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.

```

396 \@@_keys_define_code:nmn {fontspec-opentype} { Language / Default }
397 {
398     \tl_set:Nn \l_@@_lang_tl {dflt}
399     \int_zero:N \l_@@_language_int
400     \tl_gset:Nn \g_@@_single_feat_tl { language=dflt }
401 }

```

5 Backwards compatibility

```

402 \cs_new:Nn \@@_ot_compat:nm
403 {
404     \aliasfontfeatureoption {#1} {#2off} {No#2}
405 }
406 \@@_ot_compat:nm {Ligatures}    {Rare}
407 \@@_ot_compat:nm {Ligatures}    {Required}
408 \@@_ot_compat:nm {Ligatures}    {Common}
409 \@@_ot_compat:nm {Ligatures}    {Discretionary}
410 \@@_ot_compat:nm {Ligatures}    {Contextual}
411 \@@_ot_compat:nm {Ligatures}    {Historic}
412 \@@_ot_compat:nm {Numbers}      {SlashedZero}
413 \@@_ot_compat:nm {Contextuals}  {Swash}
414 \@@_ot_compat:nm {Contextuals}  {Alternate}
415 \@@_ot_compat:nm {Contextuals}  {WordInitial}
416 \@@_ot_compat:nm {Contextuals}  {WordFinal}
417 \@@_ot_compat:nm {Contextuals}  {LineFinal}
418 \@@_ot_compat:nm {Contextuals}  {Inner}
419 \@@_ot_compat:nm {Diacritics}   {MarkToBase}
420 \@@_ot_compat:nm {Diacritics}   {MarkToMark}
421 \@@_ot_compat:nm {Diacritics}   {AboveBase}
422 \@@_ot_compat:nm {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{Deseret}{dsrt}
33 \newfontscript{Devanagari}{dev2,deva}
34 \newfontscript{Dives-Akuru}{diak}
35 \newfontscript{Dogra}{dogr}
36 \newfontscript{Duployan}{dupl}
37 \newfontscript{Egyptian-Hieroglyphs}{egyp}
38 \newfontscript{Elbasan}{elba}
39 \newfontscript{Elymaic}{elym}
40 \newfontscript{Ethiopic}{ethi}
41 \newfontscript{Georgian}{geor}
42 \newfontscript{Glagolitic}{glag}
43 \newfontscript{Gothic}{goth}
44 \newfontscript{Grantha}{gran}
```



```

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

```

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

```

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

```

For convenience or backwards compatibility:

```

165 \newfontscript{CJK}{hani}
166 \newfontscript{Kana}{kana}
167 \newfontscript{Maths}{math}
168 \newfontscript{N'ko}{nko~}
169 \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{Bhojपुरi}{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{Tuviv}{TUV}
357 \newfontlanguage{Twii}{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 X_YTeX.

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

1.4 Contextuals

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

1.5 Diacritics

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

1.6 Vertical position

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

1.7 Fractions

```
64 \@@_define_aat_feature_group:n {Fractions}
65 \@@_define_aat_feature:nmmm {Fractions} {On} {11} {1}
66 \@@_define_aat_feature:nmmm {Fractions} {Off} {11} {0}
67 \@@_define_aat_feature:nmmm {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 55 \DeclareDocumentCommand \UndeclareSymbol {m}
\UndeclareCommand 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 L^AT_EX'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\plus1mu\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}

```

($3A_{16} = 58_{10}$) 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 \LaTeX 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/additions to the NFSS

`\strong` `\strong` $\langle text \rangle$

`\strongenv` `\begin{strongenv}` $\langle text \rangle$ `\end{strongenv}`

Typesets text in the ‘strong’ font. NFSS series equivalent to `\emph`. Can be nested.

`\strongfontdeclare` `\strongfontdeclare` $\langle comma-separated\ font\ switch\ declarations \rangle$

Define the behaviour of nested `\strong` commands.

`\strongreset` `\renewcommand` `\strongreset` $\langle font\ switch\ declarations \rangle$

Define the behaviour when a `\strong` command is nested deeper than the definitions provided by `\strongfontdeclare`. By default this is $\langle empty \rangle$ — i.e., bold on top of bold remains bold. In certain circumstances it may be appropriate to reset to a default state.

2 Implementation

1 $\langle *fontspec \rangle$

2.1 Italic small caps and so on

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

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

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

L^AT_EX’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

```

2.2 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 129.)

`\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 129.)

\strong

```

72 \DeclareTextFontCommand{\strong}{\strongenv}

```

(End of definition for \strong. This function is documented on page 129.)

\strongreset

```

73 \cs_set:Npn \strongreset {}

```

(End of definition for \strongreset. This function is documented on page 129.)

\reset@font Ensure nesting resets when necessary:

```

74 \cs_set_protected:Npn \reset@font
75   {
76     \normalfont
77     \int_zero:N \l_@@_strong_int
78   }

```

(End of definition for \reset@font.)

2.3 Defaults

```

79 \strongfontdeclare{\bfseries}

```

```

80 </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
@@ commands:	
\@@_DeclareFontShape:nnnnnn	535, 542, <u>552</u> , 552, 561, 570, 583, 591, 604
\g_@@_OT_features_prop	20, 22, 77
\@@_add_nfssfont:nnnn	311, 336, 337, 338, 339, 340, 341, 342, 343, <u>357</u> , 357
\@@_aff_error:n	11, 387, <u>428</u> , 460
\l_@@_alias_bool	25, 207, 214, 220, 225, 232, 252
\l_@@_all_features_clist	25, 57, 120, 130, 144, 195, 302
\g_@@_all_keyval_modules_clist	1, 51, 209, 227
\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, 298, 299, 300, 303, 306
\l_@@_atsui_bool	7, 10, 234, 376, 385, 678
\l_@@_basename_tl	44, 10, 95, 354
\l_@@_bf_series_seq	49, 157, 169, 172
\g_@@_bfmathrm_tl	72, 73, 82, 87, 88, 89, 126
\@@_calc_scale:n	318, 319, <u>331</u> , 331
\@@_calc_scale:nn	320, <u>352</u> , 352
\g_@@_cfg_bool	1, 9, 11, 19
\l_@@_check_bool	5, 58, 59, 206, 211, 217, 228
\@@_check_lang:Nn	132
\@@_check_lang:NnTF	97, <u>132</u> , 371
\@@_check_lang:Nnn	136
\@@_check_lang:NnnTF	110, <u>132</u> , 134
\@@_check_ot_feat:Nn	180
\@@_check_ot_feat:NnTF	50, 71, <u>180</u> , 670
\@@_check_ot_feat:Nnnn	185
\@@_check_ot_feat:NnnnTF	67, <u>180</u> , 182
\@@_check_script:Nn	86
\@@_check_script:NnTF	80, <u>86</u> , 304, 321, 339, 386
\@@_combo_sc_shape:n	543, 546, 592, 631, 639
\@@_construct_font_call:nn	157, 159, 163, 166, <u>172</u> , 177, 179, 305, 423, 424, 446, 529
\@@_construct_font_call:nnnnnn	172, 181
\l_@@_curr_bfname_tl	98, 167, 177, 180, 182, 217
\l_@@_curr_fontname_tl	97, 348, 349
\g_@@_curr_series_tl	96, 156, 171, 175, 180, 182, 217, 737
\@@_declare_shape:nnnn	434, <u>454</u> , 454, 472
\@@_declare_shape_loginfo:nn	470, <u>612</u> , 612
\@@_declare_shape_slanted:nn	468, <u>562</u> , 562
\@@_declare_shapes_bx:nn	469, <u>574</u> , 574
\@@_declare_shapes_normal:nn	466, <u>533</u> , 533
\@@_declare_shapes_smcaps:nn	467, <u>538</u> , 538
\g_@@_default_fontopts_clist	50, 122, 132
\@@_define_aat_feature:nnnn	2, 3, 4, 5, 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, 186

\@@_define_aat_feature_group:n .	\@@_extract_all_features:n ...
..... 1, 1, 1, 29, 35, 44, 55, 24, 115
59, 64, 68, 80, 91, 100, 108, 116, 125, 181	\l_@@_fake_ambolden_tl
\@@_define_opentype_feature:nnnnn 134, 644, 647, 661
..... 7, 16, 26, 29, 44,	\l_@@_fake_slant_tl . 133, 639, 666, 669
45, 54, 55, 55, 56, 60, 61, 74, 90, 106,	\l_@@_family_fontopts_clist
118, 124, 125, 126, 128, 133, 134, 135, 56, 126, 127, 133
138, 139, 140, 142, 166, 167, 170, 190, 196	\g_@@_family_int_prop ... 82, 279, 285
\@@_define_opentype_feature_-	\l_@@_family_label_tl
group:n 126, 128, 132, 151, 165
6, 12, 12,	\@@_feat_off:n
28, 54, 73, 89, 105, 117, 127, 137, 141, 50, 55
169, 189, 191, 207, 216, 235, 249, 271, 281	\@@_feat_prop_add:nn 1, 2, 3, 4, 5, 16, 28
\@@_define_opentype_onoffreset:nnnnn	\@@_feat_reset:n
..... 13, 14, 15, 16, 17, 51, 56, 61
18, 27, 46, 48, 49, 50, 50, 51, 52, 52,	\@@_find_autofonts:
53, 64, 65, 66, 67, 68, 72, 83, 84, 85, 293, 313, 313
86, 87, 88, 99, 100, 101, 102, 103, 104,	\l_@@_firsttime_bool
113, 114, 115, 116, 123, 136, 155, 156, 1, 38, 209, 245, 274, 384,
157, 158, 159, 160, 161, 162, 163, 164,	480, 504, 517, 529, 591, 637, 659, 689, 730
165, 168, 181, 182, 183, 184, 185, 186,	\@@_font_is_file: 29, 49, 68, 176, 189, 194
187, 188, 200, 201, 202, 203, 204, 205,	\@@_font_is_name:
206, 208, 209, 210, 211, 212, 213, 214, 215 189, 189, 731
\@@_define_opentype_onreset:nnnnn	\l_@@_font_path_tl 28, 101, 198, 203, 732
..... 58, 58	\@@_font_suppress_not_found_-
\@@_define_opentype_variation_-	error:
axis:nn 5, 9, 9, 38, 270
1, 1, 567, 574, 575	\l_@@_fontcfg_bool ... 12, 13, 18, 22, 96
\g_@@_defined_shapes_tl	\l_@@_fontface_cs_tl .. 17, 71, 150,
..... 93, 198, 614, 736	155, 160, 161, 164, 165, 168, 169, 304,
\l_@@_defining_encoding_bool .. 3,	321, 339, 340, 371, 386, 445, 451, 670, 681
9, 15, 21, 27, 27, 50, 52, 57, 63, 69, 75	\l_@@_fontfeat_bf_clist
\l_@@_disable_defaults_bool 66, 215, 337, 662
..... 24, 118, 156	\l_@@_fontfeat_bfit_clist
\l_@@_em_int 68, 225, 341, 646, 648, 668, 670
40	\l_@@_fontfeat_bfsl_clist 70, 233, 342
\g_@@_em_normalise_slant_bool ... 29	\l_@@_fontfeat_bfsw_clist 72, 241, 343
\g_@@_em_prop	\l_@@_fontfeat_clist . 61, 151, 223, 275
79	\l_@@_fontfeat_curr_clist
\l_@@_em_switch_tl 62, 489, 498, 511
118	\l_@@_fontfeat_it_clist
\l_@@_em_tmp_tl 67, 221, 338, 640
123	\l_@@_fontfeat_sc_clist . 73, 247, 489
\l_@@_emdef_int	\l_@@_fontfeat_sl_clist . 69, 229, 339
41	\l_@@_fontfeat_sw_clist . 71, 237, 340
\l_@@_emshape_query_tl	\l_@@_fontfeat_up_clist
117 65, 211, 253, 336
\@@_error:n	\g_@@_fontid_family_prop 81, 259, 287
1, 10, 480	\l_@@_fontid_tl
\@@_error:nn 44, 25, 27, 102, 255, 259, 267
2, 3, 4, 10, 14, 16,	\l_@@_fontname_bf_tl
22, 28, 58, 64, 70, 76, 161, 447, 770, 789 90, 136, 177, 318, 324, 337, 663
\@@_error:nnn	\l_@@_fontname_bfit_tl
4, 456 91, 138, 188, 317, 318, 319, 341, 649, 671
\l_@@_ext_filename_tl	\l_@@_fontname_bfsl_tl
..... 99, 99, 100, 103, 105, 108, 109, 110 140, 192, 332, 342
\l_@@_extension_tl	\l_@@_fontname_bfsw_tl .. 142, 196, 343
..... 14, 33, 38, 44, 67, 87, 100, 121, 183	
\l_@@_extensions_clist 54, 62, 76, 257	
\l_@@_external_bool 26, 39, 49, 197, 407	
\l_@@_external_kpse_bool . 30, 48, 198	
\@@_extract_all_features:	
..... 115	

\l_@@_fontname_it_tl	265, 269, 273, 277, 281, 282, 283, 284,
..... 89, 137, 143, 317, 329, 338, 641	285, 286, 287, 291, 295, 314, 325, 382,
\l_@@_fontname_sc_tl 143, 206, 493, 505	396, 408, 429, 433, 437, 461, 523, 540,
\l_@@_fontname_sl_tl 139, 148, 332, 339	544, 550, 562, 569, 576, 601, 605, 677, 681
\l_@@_fontname_sw_tl 141, 152, 340	\l_@@_keys_leftover_clist
\l_@@_fontname_tl ... 103, 155, 157, 161 59, 145, 148, 149,
\l_@@_fontname_up_tl . 44, 48, 9, 13,	150, 224, 225, 229, 230, 233, 235, 239, 240
32, 133, 135, 147, 157, 159, 161, 163, 166	\@@_keys_set_known:nnN
\@@_fontname_wrap:n	60, 66, 143, 148, 150, 223, 225, 363, 432
..... 49, 174, 175, 191, 192, 200, 203	\l_@@_lang_name_tl
\l_@@_fontopts_clist 117, 135, 148, 149, 213, 215
.. 55, 123, 124, 134, 434, 442, 443, 444	\l_@@_lang_tl
\g_@@_fontopts_prop	48, 147, 182, 311, 373, 385, 398, 655, 664
74, 101, 123, 126, 135, 138, 142, 143, 442	\l_@@_language_int
\@@_format_axis:nn	34,
707, 721	45, 198, 199, 203, 209, 309, 374, 389, 399
\@@_get_features:Nn	\l_@@_leftover_clist 58, 432, 434
63, 219	\@@_load_external_fontoptions:N
\@@_get_features:n ... 37, 219, 276, 518 22, 94, 94, 441
\@@_get_variations:	\@@_load_font:
..... 62, 306, 519, 530, 706, 711	35, 153, 153
\l_@@_graphite_bool .. 11, 234, 379, 397	\@@_load_fontname:Nn 433, 437, 484, 505
\l_@@_harfbuzz_bool	\@@_lua_function:nn
10, 79, 102	71, 72
\c_@@_hexcol_tl	\@@_lua_function:nnn
160, 247, 750	71, 73
\l_@@_hexcol_tl 152, 246, 248,	\@@_lua_function:nnnn 71, 74, 174
249, 466, 471, 475, 490, 495, 499, 514, 750	\@@_lua_function:nnnnn ... 71, 75, 227
\l_@@_hyphenchar_tl	\@@_main_DeclareFontExtensions:n
..... 151, 448, 449, 451, 454 122, 255, 255, 259
\@@_if_autofont:nn	\@@_main_IfFontFeatureActiveTF:nnn
420 126, 260
\@@_if_autofont:nnTF	\@@_main_addfontfeatures:n 82, 86, 147
413	\@@_main_aliasfontfeature:nn 106, 204
\@@_if_detect_external:n	\@@_main_aliasfontfeatureoption:nnn
73 110, 223
\@@_if_detect_external:nTF 16, 73, 176	\@@_main_fontspec:nn
\@@_if_font_feature:n	1, 1, 3
266	\@@_main_liningnums:n
\@@_if_font_feature:nTF	137, 295
264	\@@_main_newAATfeature:nnnn .. 94, 178
\@@_if_merge_shape:n	\@@_main_newfontface:NnnN
21 59, 63, 67, 71, 115, 115
\@@_if_merge_shape:nTF	\@@_main_newfontfamily:NnnN
192 43, 47, 51, 55, 102, 102, 117
\@@_info:n	\@@_main_newfontfeature:nn ... 90, 171
8, 133, 323, 329	\@@_main_newopentypefeature:nnn
\@@_info:nn 98, 102, 188
9, 415	\@@_main_oldstylenums:n 132, 288
\@@_info:nnn	\@@_main_setboldmathrm:nn ... 27, 70
10, 296	\@@_main_setmainfont:nn 8, 24, 39
\@@_init:	\@@_main_setmathrm:nn
6, 175, 271, 726, 726	23, 64
\@@_init_fontface: 222, 743, 743	\@@_main_setmathsf:nn
\@@_init_ttc:n	31, 76
21, 85, 85	\@@_main_setmathtt:nn
\g_@@_instance_tl 159, 607, 713, 715, 747	35, 82
\@@_int_mult_truncate:Nn . 67, 67, 526	\@@_main_setmonofont:nn
\@@_iv_str_to_num:Nn	18, 51
97, 146, 147, 195, 199, 200, 773, 774, 779	\@@_main_setsansfont:nn
\@@_iv_str_to_num:w 783, 784, 786	13, 38
\@@_keys_define_code:nnn ... 7, 13,	\@@_make_AAT_feature:nn 9, 12, 12, 76, 87
16, 20, 24, 35, 36, 45, 46, 51, 109, 114,	
119, 126, 131, 135, 146, 150, 154, 159,	
186, 190, 194, 198, 209, 213, 219, 223,	
227, 231, 235, 239, 243, 250, 255, 261,	

<code>\@@_make_AAT_feature_string:Nnn</code>	<code>\c_@@_opacity_tl</code>
<code>\@@_make_AAT_feature_string:NnnTF</code>	<code>\l_@@_opacity_tl</code>
<code>\@@_make_OT_feature:nnn</code>	<code>\l_@@_optical_size_tl</code>
<code>\@@_make_font_shapes:Nnnnn</code>	<code>\l_@@_options_tl</code>
<code>\@@_make_ot_smallcaps:TF</code>	<code>\l_@@_ot_bool</code>
<code>\@@_make_smallcaps:TF</code>	<code>\@@_ot_compat:nn</code>
<code>\l_@@_mapping_tl</code>	<code>\@@_ot_validate_tag:n</code>
<code>\g_@@_math_bool</code>	<code>\@@_ot_validate_tag:w</code>
<code>\g_@@_math_euler_bool</code>	<code>\@@_ot_validate_tag_aux:w</code>
<code>\g_@@_math_lucida_bool</code>	<code>\g_@@_pkg_euler_loaded_bool</code>
<code>\g_@@_mathrm_tl</code>	<code>\c_@@_postadjust_tl</code>
<code>\g_@@_mathsf_tl</code>	<code>\l_@@_postadjust_tl</code>
<code>\g_@@_mathtt_tl</code>	<code>\l_@@_pre_feat_sclist</code>
<code>\@@_merge_default_shapes:</code>	<code>\@@_prepare_features:</code>
<code>\l_@@_mm_bool</code>	<code>\l_@@_prev_unicode_name_tl</code>
<code>\l_@@_mode_tl</code>	<code>\l_@@_primitive_font</code>
<code>\@@_msg_new:nn</code>	<code>\@@_primitive_font_current_name:</code>
<code>\@@_msg_new:nnn</code>	<code>\@@_primitive_font_get_name:N</code>
<code>\l_@@_never_check_bool</code>	<code>\@@_primitive_font_glyph_if_</code>
<code>\g_@@_nfss_enc_tl</code>	<code>\@@_primitive_font_glyph_if_</code>
<code>\l_@@_nfss_fam_tl</code>	<code>\@@_primitive_font_gset:Nnn</code>
<code>\g_@@_nfss_family_tl</code>	<code>\@@_primitive_font_gset:NnnTF</code>
<code>\l_@@_nfss_prop</code>	<code>\@@_primitive_font_gset:Onn</code>
<code>\l_@@_nfss_sc_tl</code>	<code>\@@_primitive_font_gset:p:NnnTF</code>
<code>\l_@@_nfss_tl</code>	<code>\@@_primitive_font_if_exist:n</code>
<code>\l_@@_nfssfont_prop</code>	<code>\@@_primitive_font_if_exist:nTF</code>
<code>\l_@@_nobf_bool</code>	<code>\@@_primitive_font_if_null:N</code>
<code>\l_@@_noit_bool</code>	<code>\@@_primitive_font_if_null:NTF</code>
<code>\l_@@_nosc_bool</code>	<code>\@@_primitive_font_if_null_p:N</code>
	<code>\@@_primitive_font_set:Nnn</code>

\@@_primitive_font_set:NnnTF .	32, 158	\@@_set_family:NnnN	147, 157, 158
\@@_primitive_font_set:NnnTTF . .	21	\@@_set_font_default_features:nnn	77, 124, 124
\@@_primitive_font_set:Onn	31	\@@_set_font_dimen:NnN	339, 340, 366, 366
\@@_primitive_font_set:OnnTF .	32, 445	\@@_set_font_type:N	9, 27, 38, 64, 77, 90, 107, 120, 135, 164, 371, 371
\@@_primitive_font_set_hyphenchar:Nn	52, 52, 442, 454	\@@_set_fontface:NNnnN . .	162, 170, 171
\@@_primitive_font_set_p:NnnTF . .	21	\@@_set_scriptlang:	36, 206, 206
\l_@@_proceed_bool	31, 67, 74, 80	\@@_setboldmathrm_hook:nn . . .	74, 94
\@@_process_ext:N	57, 60, 355	\@@_setmainfont_hook:nn	35, 88
\l_@@_punctspace_adjust_tl	156, 163, 414, 419, 424, 753	\@@_setmathrm_hook:nn	68, 91
\g_@@_rawfeatures_sclist	62, 164, 279, 306, 519, 530, 693, 702, 745	\@@_setmathsf_hook:nn	80, 92
\g_@@_rawvariations_prop	6, 83, 603, 717, 721, 746	\@@_setmathtt_hook:nn	86, 93
\@@_remove_clashing_featstr:n	34, 78, 696, 696, 705	\@@_setmonofont_hook:nn	61, 90
\l_@@_renderer_tl	55, 60, 64, 65, 90, 185, 386, 394, 398, 735	\@@_setsansfont_hook:nn	48, 89
\l_@@_rmfamily_encoding_tl	9, 15, 21, 32, 169	\@@_setup_nfss:Nnnn .	485, 510, 514, 514
\l_@@_rmfamily_family_tl .	30, 31, 166	\@@_setup_single_size:nn	461, 473, 473
\@@_sanitise_fontname:Nn	8, 9, 10, 53, 53, 89, 90, 91, 131, 440	\l_@@_sffamily_encoding_tl	10, 17, 22, 45, 170
\@@_save_family:nn	42, 290, 290	\l_@@_sffamily_family_tl .	43, 44, 167
\@@_save_family_needed:n	251	\@@_shape_merge:nn	5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 24, 29, 194, 548, 549
\@@_save_family_needed:nTF . . .	40, 251	\l_@@_shaper_tl .	86, 91, 92, 96, 105, 662
\@@_save_fontid_family:nn	267, 277, 289	\g_@@_single_feat_tl	65, 93, 94, 268, 280, 282, 284, 311, 325, 346, 375, 390, 400, 691
\@@_save_fontinfo:n	292, 298, 298	\l_@@_size_tl	112, 275, 475, 480, 481, 516, 528
\l_@@_saved_fontname_tl	105, 459, 476	\l_@@_sizedfont_tl	113, 279, 476, 484, 486
\l_@@_scale_tl	150, 206, 322, 327, 328, 342, 350, 356, 358, 360, 364, 521, 523, 528, 748	\l_@@_sizefeat_clist	52, 53, 252, 257, 362, 368
\l_@@_script_int	33, 42, 94, 147, 149, 155, 200, 203, 209, 308, 309, 324, 344	\l_@@_sizing_leftover_clist	60, 479, 485, 511
\l_@@_script_name_tl	112, 135, 145, 145, 146, 211, 213, 214, 307, 323, 342, 353, 388	\l_@@_smcp_shape_tl	116, 194, 197, 200, 203
\l_@@_script_tl	47, 95, 134, 144, 182, 308, 310, 320, 321, 325, 343, 386, 654, 663	\@@_strip_leading_sign:Nw . .	777, 780
\l_@@_scriptlang_exist_bool . . .	28, 301, 310, 315, 336, 345, 350, 368, 376, 380	\@@_strip_plus_minus:n	197, 199
\@@_select_font_family:nn	1, 1, 52, 152, 157, 160, 166	\@@_strip_plus_minus_aux:Nq .	199, 200
\@@_set_autofont:Nnn	317, 318, 319, 324, 329, 332, 405, 405	\l_@@_strnum_int	35, 97, 103, 146, 157, 195, 210, 309, 324, 344, 374, 389
\@@_set_default_features:nn	76, 119, 119	\l_@@_strong_int	42, 55, 58, 59, 62, 64, 67, 77
\@@_set_faces:	295, 334, 334	\g_@@_strong_prop	37, 45, 46, 56, 64, 80
\@@_set_faces_aux:nnnn	344, 346	\l_@@_strong_switch_tl . . .	64, 65, 119
		\l_@@_strong_tmp_tl	56, 58, 124
		\l_@@_strongdef_int	38, 43, 45, 46, 47
		\@@_swap_plus_minus:n	78, 83
		\@@_swap_plus_minus_aux:Nq . . .	83, 84
		\l_@@_test_font	158, 164

<code>\l_@@_tfm_bool</code>	6, 375, 382	<code>\aliasfontfeatureoption</code>	69, 70, 71, 108, 223, 404
<code>\l_@@_this_feat_clist</code>	64, 299, 307, 312	<code>\AtBeginDocument</code>	123, 53, 128, 137
<code>\l_@@_this_font_tl</code>	114, 258, 259, 263, 297, 306, 312, 359, 365, 368	<code>\author</code>	35
<code>\@@_tl_new_if_free:N</code>	146, 153	B	
<code>\l_@@_tmp_int</code>	36, 525, 526, 535, 536	<code>\bar</code>	22
<code>\l_@@_tmp_tl</code>	41, 42, 44, 45, 93, 94, 115, 116, 117, 118, 120, 123, 124, 130, 131, 135, 138, 138, 139, 142, 143, 153, 172, 173, 174, 225, 226, 227, 259, 261, 265, 266, 267, 279, 281, 282, 284, 285, 286, 356, 360, 363	<code>\bfdefault</code>	52, 81, 28, 41, 54, 78, 81, 84, 88, 91, 92, 171, 172, 175, 337, 341, 342, 343, 578, 579, 585, 593, 620, 624, 625, 626, 634, 636, 638
<code>\l_@@_tmpa_bool</code>	23, 78, 81, 83	<code>\bfseries</code>	79
<code>\l_@@_tmpa_dim</code>	46, 339, 344	<code>\boldmath</code>	28
<code>\l_@@_tmpa_font</code>	423, 425	bool commands:	
<code>\l_@@_tmpa_fp</code>	44	<code>\bool_gset_false:N</code>	3, 98, 100, 101, 102, 103, 104, 105, 106, 107, 112, 113, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130
<code>\l_@@_tmpa_int</code>	37, 150, 152, 155, 160, 163	<code>\bool_gset_true:N</code>	2, 9, 13, 14, 15
<code>\l_@@_tmpa_tl</code>	28, 29, 53, 121	<code>\bool_if:NTF</code>	1, 3, 8, 9, 10, 15, 21, 27, 28, 34, 39, 39, 40, 49, 57, 63, 65, 69, 75, 78, 80, 83, 89, 91, 96, 108, 110, 118, 119, 121, 131, 136, 139, 166, 175, 187, 209, 217, 220, 227, 228, 245, 252, 315, 322, 327, 350, 380, 384, 407, 480, 491, 504, 508, 517, 529, 579, 584, 591, 637, 652, 659, 675, 678, 689
<code>\l_@@_tmpb_dim</code>	47, 340, 345	<code>\bool_if:nTF</code>	92, 142, 190, 234, 315, 564, 576, 589, 598, 763, 782
<code>\l_@@_tmpb_font</code>	424, 425	<code>\bool_lazy_and:nnTF</code>	23, 198
<code>\l_@@_tmpb_fp</code>	45	<code>\bool_new:N</code>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 19, 20, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
<code>\l_@@_tmpb_int</code>	38, 148, 152, 160	<code>\bool_set_false:N</code>	18, 22, 38, 52, 59, 74, 78, 100, 110, 116, 142, 151, 166, 205, 206, 207, 225, 274, 301, 336, 368, 375, 376, 377, 378, 379, 642, 664, 729
<code>\l_@@_tmpb_tl</code>	34, 39, 42, 46, 50, 53, 122, 135, 136, 137, 138	<code>\bool_set_true:N</code>	13, 26, 27, 48, 50, 58, 67, 79, 81, 102, 104, 139, 156, 159, 163, 197, 202, 211, 214, 232, 272, 273, 310, 345, 376, 382, 385, 389, 393, 397, 402, 502, 730
<code>\l_@@_tmpc_dim</code>	48	<code>\bool_until_do:nn</code>	101, 152, 207
<code>\l_@@_tmpc_int</code>	39, 154, 157	C	
<code>\@@_trace:n</code>	11	char commands:	
<code>\l_@@_ttc_index_tl</code>	115, 123, 124, 128, 129, 184, 734	<code>\char_set_catcode_ignore:n</code>	236
<code>\l_@@_ttfamily_encoding_tl</code>	11, 19, 23, 58, 171	<code>\char_set_catcode_space:n</code>	18
<code>\l_@@_ttfamily_family_tl</code>	56, 57, 168	<code>\check</code>	24
<code>\@@_update_featstr:n</code>	19, 80, 175, 244, 248, 249, 435, 564, 571, 586, 613, 621, 629, 679, 683, 686, 686	clist commands:	
<code>\@@_warning:n</code>	5, 15, 40, 122, 554, 558	<code>\clist_clear:N</code>	124, 127, 275, 443
<code>\@@_warning:nn</code>	6, 22, 68, 73, 103, 167, 221, 253, 383, 445, 481, 505, 518, 530, 592		
<code>\@@_warning:nnn</code>	7, 43, 184, 194, 318		
<code>\l_@@_wordspace_adjust_tl</code>	157, 163, 392, 400, 752		
<code>\</code>	17, 26, 32, 36, 37, 37, 38, 38, 101, 102, 103, 168, 185, 195, 196, 197, 220, 225, 231, 232, 233, 616, 630, 644, 645		
<code>_</code>	33		
A			
<code>\acute</code>	18		
<code>\addfontfeature</code>	31, 51, 84, 100, 291, 298		
<code>\addfontfeatures</code>	80, 80, 147		
<code>\aliasfontfeature</code>	34, 90, 104, 204, 234, 248, 522		

<code>\dot</code>	26	<code>\fontdimen</code>	86, 87, 368, 394,
<code>\DTX</code>	3, 4, 5, 6, 7, 8, 9, 10, 11,		395, 396, 402, 403, 404, 415, 420, 425
	12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23	<code>\fontdimen8</code>	86
E			
<code>\else</code>	17, 106, 213, 795, 796	<code>\fontencoding</code> .	4, 9, 10, 11, 15, 17, 19, 110, 335
else commands:		<code>\fontfamily</code>	
<code>\else:</code>	48	..	30, 14, 15, 16, 17, 18, 19, 109, 164, 336
<code>\emfontdeclare</code>	34	<code>\fontname</code>	42, 56
<code>\emph</code>	129	<code>\fontspec</code>	24, 31, 44, 1
<code>\EncodingAccent</code>	7	fontspec commands:	
<code>\EncodingCommand</code>	1	<code>\fontspec_calc_scale:n</code>	84, 86
<code>\EncodingComposite</code>	19	<code>\fontspec_calc_scale:nn</code>	84
<code>\EncodingCompositeCommand</code>	25	<code>\fontspec_complete_fontname:Nn</code> .	
<code>\encodingdefault</code> 85, 21, 22, 23, 34, 47, 60, 335		133, 143, 148, 152,
<code>\EncodingSymbol</code>	13	167, 188, 192, 196, 206, 279, 348, 351, 351
<code>\endinput</code>	8, 13, 22, 28	<code>\g_fontspec_default_fontopts_tl</code> .	31
exp commands:		<code>\fontspec_default_script:nn</code> .	331, 358
<code>\exp_after:wN</code>	478	<code>\g_fontspec_encoding_tl</code>	44,
<code>\exp_args:NNNx</code>	348, 362	48, 49, 51, 52, 55, 56, 74, 75, 77, 78,
<code>\exp_args:Nnnx</code>	196	79, 80, 81, 84, 85, 87, 88, 89, 91, 92, 738
<code>\exp_args:Nnx</code>	54, 55, 56, 60, 61	<code>\l_fontspec_family_tl</code> 44, 50, 84, 154, 167	
<code>\exp_args:No</code>	17,	<code>\l_fontspec_feature_string_tl</code> .	19, 53
.....	71, 103, 109, 304, 339, 371, 451, 670, 680	<code>\l_fontspec_font</code> .	44, 150, 155, 161, 168
<code>\exp_args:Noo</code>	321, 386	<code>\fontspec_font_if_exist:n</code>	172
<code>\exp_args:NV</code>	284	<code>\fontspec_font_if_exist:nTF</code>	
<code>\exp_args:Nx</code>	79, 461	36, 172, 181
<code>\exp_args:Nxx</code>	186	<code>\l_fontspec_fontname_tl</code>	44,
<code>\exp_last_unbraced:No</code> ..	31, 32, 33, 34	48, 8, 12, 22, 25, 31, 86, 119, 123, 124,
<code>\exp_not:N</code>	21, 98,	129, 134, 144, 147, 206, 305, 318, 319,
.....	107, 109, 110, 111, 252, 253, 256, 257,	324, 329, 336, 433, 451, 456, 456, 459,
.....	265, 266, 279, 280, 439, 617, 631, 644, 645	<code>\fontspec_gset_family:Nnn</code> ...	36,
<code>\exp_not:n</code>	65, 75, 221, 263, 616, 630	66, 67, 72, 73, 78, 79, 84, 85, 146, 157
<code>\expandafter</code>	28	<code>\fontspec_gset_fontface:NNnn</code> ...	
F			
<code>\familydefault</code>	33, 46, 59, 336	36, 160, 170
<code>\fi</code>	19, 28, 29, 32, 42, 55,	<code>\fontspec_if_aat_feature:nn</code>	5
	98, 108, 110, 116, 215, 390, 399, 795, 796	<code>\fontspec_if_aat_feature:nnTF</code> ..	36, 5
fi commands:		<code>\fontspec_if_current_feature:n</code> .	182
<code>\fi:</code>	50	<code>\fontspec_if_current_feature:nTF</code>	
file commands:		37, 182, 284
<code>\file_if_exist:nTF</code>	103, 109	<code>\fontspec_if_current_language:n</code> 131	
<code>\file_input:n</code>	105, 110	<code>\fontspec_if_current_language:nTF</code>	
<code>\filedate</code>	55	37, 131
<code>\fileversion</code>	55	<code>\fontspec_if_current_script:n</code> ..	116
<code>\fmtname</code>	28	<code>\fontspec_if_current_script:nTF</code>	
<code>\font</code>	44, 3, 7, 9, 12, 27, 38, 50, 64,	37, 116
	67, 77, 80, 90, 97, 107, 110, 114, 120,	<code>\fontspec_if_feature:n</code>	34
	135, 171, 224, 339, 373, 394, 395, 396,	<code>\fontspec_if_feature:nnn</code>	60
	402, 403, 404, 415, 420, 425, 442, 454	<code>\fontspec_if_feature:nnnTF</code>	37, 60
		<code>\fontspec_if_feature:nTF</code>	37, 34
		<code>\fontspec_if_fontspec_font:</code>	1

<code>\fontspec_if_fontspec_font:TF</code>	36, 1, 7, 25, 36, 62, 75, 88, 105, 118, 133, 150
<code>\fontspec_if_language:n</code>	86
<code>\fontspec_if_language:nn</code>	103
<code>\fontspec_if_language:nnTF</code>	37, 103
<code>\fontspec_if_language:nTF</code>	37, 86
<code>\fontspec_if_opentype:</code>	23
<code>\fontspec_if_opentype:TF</code>	36, 23
<code>\fontspec_if_script:n</code>	73
<code>\fontspec_if_script:nTF</code>	37, 73
<code>\fontspec_if_small_caps:</code>	190
<code>\fontspec_if_small_caps:TF</code>	37, 190
<code>\fontspec_maybe_setup_maths:</code>	94, 94, 137
<code>\fontspec_new_lang:nn</code>	118, 364
<code>\fontspec_new_script:nn</code>	114, 296
<code>\fontspec_parse_colour:niii</code>	478, 502, 512
<code>\fontspec_parse_cv:w</code>	252, 265
<code>_fontspec_parse_wordspace:w</code>	385, 388, 388
<code>\fontspec_select:nn</code>	52, 52
<code>\fontspec_set_family:Nnn</code>	36, 3, 30, 43, 56, 104, 146, 158, 159
<code>\fontspec_set_fontface:NNnn</code>	36, 160, 171
<code>\fontspec_setup_maths:</code>	1, 4, 134
fontspec internal commands:	
<code>_fontspec_calc_scale:n</code>	355, 357
<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>	58
<code>\FontspecSetCheckBoolTrue</code>	58
fp commands:	
<code>\fp_eval:n</code>	328, 344, 360
<code>\fp_new:N</code>	44, 45
G	
<code>\g</code>	125
<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, 152, 174, 269, 290, 297, 333, 354, 422, 431, 555
<code>\group_end:</code>	33, 41, 42, 48, 49, 163, 178, 179, 277, 293, 300, 349, 363, 426, 427, 435, 558
H	
<code>\hat</code>	123, 25
<code>\hbar</code>	71
I	
<code>\IfBooleanTF</code>	121, 133
<code>\ifcase</code>	380
<code>\ifdefined</code>	27, 40, 53
<code>\IfFontExistsTF</code>	181
<code>\IfFontFeatureActiveTF</code>	124, 260
<code>\IfNoValueTF</code>	75
<code>\ifnum</code>	103, 209, 387
<code>\ifx</code>	15, 28, 30, 98, 110, 116, 795, 796
<code>\ignorespaces</code>	4, 9, 14, 19, 78, 169
<code>\InputIfFileExists</code>	3
int commands:	
<code>\int_case:nn</code>	62, 83, 88
<code>\int_case:nnTF</code>	374
<code>\int_compare:nNnTF</code>	157
<code>\int_compare:nTF</code>	32, 58, 79, 104, 300, 474, 477, 498, 501, 535, 769, 788
<code>\int_compare_p:nNn</code>	101, 152, 207
<code>\int_eval:n</code>	282
<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, 58, 69, 77, 94, 98, 105, 148, 154, 160, 202, 212, 309, 324, 344, 374, 389, 525, 791
<code>\int_to_hex:n</code>	536
<code>\int_use:N</code>	46, 55, 59, 64
<code>\int_zero:N</code>	38, 67, 77, 99, 150, 198, 205, 399
<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, 338, 341, 566, 567, 571, 600, 601, 606, 621, 624
K	
keys commands:	
<code>\l_keys_choice_int</code>	58, 62, 79, 83, 88
<code>\keys_define:nn</code>	3, 3, 7, 9, 14, 20, 22, 31, 36, 39, 53, 69, 74, 81, 92, 173, 215, 217, 228, 233, 236, 240, 247, 259, 272, 282, 291, 298, 333, 359, 366, 449, 609, 617, 625, 633, 655

<code>\keys_if_choice_exist:nnnTF</code>	183, 193	<code>\msg_new:nnn</code>	12, 15, 15, 31, 173, 177
<code>\keys_if_exist:nnTF</code>	180, 190, 211, 229, 237, 244	<code>\msg_new:nnnn</code>	13, 17
<code>\l_keys_key_tl</code>	118, 123, 128, 133	<code>\msg_redirect_module:nnn</code>	17, 18, 22, 23, 27, 28
<code>\keys_set:nn</code>	8, 17, 31, 41, 78, 111, 116, 214, 215, 216, 234, 241, 248, 556	<code>\msg_redirect_name:nnn</code>	555
<code>\keys_set_groups:nnn</code>	444	<code>\msg_trace:nn</code>	11
<code>\keys_set_known:nn</code>	19, 230, 235, 240	<code>\msg_warning:nn</code>	3, 5
<code>\keys_set_known:nnN</code>	63, 79, 153, 478	<code>\msg_warning:nnn</code>	6, 6, 13, 14
<code>\l_keys_value_tl</code>	118, 123, 128, 133	<code>\msg_warning:nnnn</code>	7
L		N	
<code>\l</code>	31, 54, 56, 68–70, 76	<code>\newAATfeature</code>	92, 178
<code>\Lambda</code>	55	<code>\NewDocumentCommand</code>	1, 6, 11, 16, 21, 25, 29, 33, 37, 41, 43, 45, 49, 53, 57, 59, 61, 65, 69, 73, 80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 135
<code>\latinencoding</code>	52, 56, 89	<code>\newfontface</code>	30, 57
<code>\liningnums</code>	35, 135, 288	<code>\newfontfamily</code>	41
lua commands:		<code>\newfontfeature</code>	32, 88, 171
<code>\lua_now:n</code>	6, 72, 73, 74, 75, 118	<code>\newfontlanguage</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, 113, 114, 115, 116, 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, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 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,
<code>\LuaLaTeX</code>	33		
M			
<code>\mathalpha</code>	18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62		
<code>\mathbf</code>	125, 60, 78, 88		
<code>\mathbin</code>	63		
<code>\mathchar</code>	71		
<code>\mathchardef</code>	29		
<code>\mathclose</code>	36, 39, 66, 68		
<code>\mathdollar</code>	70		
<code>\mathit</code>	60, 77, 84, 89		
<code>\mathopen</code>	65, 67		
<code>\mathord</code>	69, 70		
<code>\mathpunct</code>	31, 38		
<code>\mathrel</code>	37, 64		
<code>\mathring</code>	27		
<code>\mathrm</code>	28, 125, 76, 87		
<code>\mathsf</code>	79, 91		
<code>\mathtt</code>	80, 92		
<code>\mddefault</code>	74, 75, 77, 79, 80, 87, 89, 336, 338, 339, 340, 619, 621, 622, 623, 633, 635, 637		
<code>\mkern</code>	71		
msg commands:			
<code>\msg_error:nn</code>	1		
<code>\msg_error:nnn</code>	2, 3		
<code>\msg_error:nnnn</code>	4		
<code>\msg_fatal:nn</code>	21		
<code>\msg_info:nn</code>	8		
<code>\msg_info:nnn</code>	9		
<code>\msg_info:nnnn</code>	10		
<code>\msg_line_context:</code>	100		

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	
<code>\newICUfeature</code> 100	
<code>\newopentypefeature</code> 96, 188	
<code>\normalfont</code> 36, 41, 49, 62, 76	
<code>\normalsize</code> 50, 556	
<code>\nullfont</code> 15	
<code>\numexpr</code> 44	
O	
<code>\oldstylenums</code> 35, 128, 288	
<code>\Omega</code> 62	
<code>\or</code> 383, 391, 395	
P	
<code>Path</code> 24	
<code>\Phi</code> 60	
<code>\Pi</code> 57	
prg commands:	
<code>\prg_new_conditional:Nnn</code> 1, 5, 13, 21, 21, 23, 26, 26, 34, 35, 44,	60, 73, 73, 86, 86, 103, 116, 131, 132, 136, 172, 180, 182, 185, 190, 251, 266, 420
<code>\prg_return_false:</code> 3, 13, 16, 18, 20, 24, 28, 29, 30, 31, 32, 41, 49, 50, 51, 53, 57, 67, 69, 71, 80, 82, 83, 84, 93, 97, 99, 101, 110, 110, 112, 114, 125, 126, 127, 129, 134, 140, 142, 143, 144, 166, 175, 179, 183, 188, 191, 205, 208, 217, 228, 262, 282, 285, 426	
<code>\prg_return_true:</code> 3, 13, 16, 24, 28, 29, 32, 42, 47, 50, 54, 67, 80, 83, 90, 97, 110, 110, 122, 125, 134, 140, 140, 166, 175, 178, 183, 188, 188, 206, 217, 228, 268, 274, 285, 427	
<code>\ProcessKeyOptions</code> 37	
prop commands:	
<code>\prop_gclear:N</code> 37, 746	
<code>\prop_get:NnN</code> 41, 44, 47, 48, 93, 95, 123, 138, 154, 155	
<code>\prop_get:NnNTF</code> 56, 64, 84, 85, 123, 126, 135, 259, 279, 442	
<code>\prop_gput:Nnn</code> 6, 22, 46, 83, 138, 143, 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, 285, 286, 287, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 301, 302, 302, 303, 303, 304, 305, 306, 307, 308, 308, 309, 309, 310, 310, 311, 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	
<code>\prop_gput_from_keyval:Nn</code> 603	
<code>\prop_gput_if_new:Nnn</code> 45, 82	
<code>\prop_gremove:Nn</code> 142	
<code>\prop_if_empty:NnTF</code> 717	
<code>\prop_if_in:NnTF</code> 20, 101	
<code>\prop_map_function:NN</code> 721	
<code>\prop_map_inline:Nn</code> 344	
<code>\prop_new:N</code> 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 300	
<code>\prop_put:Nnn</code> 80, 81, 180, 216, 367	
<code>\providecommand</code> 2, 3, 4	
<code>\ProvideDocumentCommand</code> 55, 71	

<code>\providefontface</code>	69	<code>\settoheight</code>	371
<code>\providefontfamily</code>	53	<code>\sfdefault</code>	44, 46, 46, 100, 130
<code>\ProvidesExplFile</code>	48	<code>\sffamily</code>	7
<code>\ProvidesExplPackage</code>	43, 44, 45	<code>\shapedefault</code> 8, 17, 18, 74, 75, 78, 79, 80,	
<code>\Psi</code>	61	81, 87, 88, 91, 92, 204, 336, 337, 619, 620	
Q			
<code>\quad</code>	55	<code>\Sigma</code>	58
quark commands:		<code>\sldefault</code>	3, 10, 13,
<code>\q_nil</code> 83, 84, 199, 200, 253, 266, 758,		16, 339, 342, 567, 570, 601, 605, 622, 625	
761, 764, 765, 767, 777, 780, 783, 784, 786		<code>\space</code>	34, 39, 44, 206
<code>\q_stop</code>	385, 388	str commands:	
R			
<code>\relax</code>	29, 44, 387	<code>\c_backslash_str</code>	78
<code>\renewcommand</code>	71	<code>\c_colon_str</code>	253, 266
<code>\RenewDocumentCommand</code>	47, 63, 130	<code>\str_case:nn</code>	85, 617, 631
<code>\renewfontface</code>	61	<code>\str_case:nnTF</code>	202, 316
<code>\renewfontfamily</code>	45	<code>\str_case_e:nnTF</code>	410
<code>\RequirePackage</code>	5, 7, 12, 42, 47, 48	<code>\str_if_eq:nnTF</code> .. 33, 46, 59, 86, 87,	
<code>\rmdefault</code>	29, 125, 31, 33, 45, 99, 129	121, 124, 139, 177, 197, 246, 373, 439, 552	
<code>\rmfamily</code>	85, 7, 373	<code>\str_if_eq_p:nn</code>	
S			
scan commands:		.. 566, 567, 578, 579, 600, 601, 763, 782	
<code>\scan_stop</code>	3, 7, 46, 54	<code>\str_lowercase:n</code>	87, 121
<code>\scdefault</code>	2, 3, 4, 8, 9, 10, 11,	<code>\string</code>	22, 60, 80, 100
12, 13, 14, 17, 18, 548, 549, 550, 633, 634		<code>\strong</code>	129, 72
<code>\scitdefault</code> 2, 9, 12, 15, 16, 17, 635, 636		<code>\strongenv</code>	129, 51, 72
<code>\scsldefault</code> ... 3, 10, 13, 15, 16, 18, 637, 638		<code>\strongfontdeclare</code>	129, 35, 79
<code>\scswdefault</code>	4, 11, 14	<code>\strongreset</code>	129, 42, 68, 73
<code>\select</code>	19	<code>\suppressfontnotfounderror</code>	11
<code>\selectfont</code>	5, 111, 164, 337	<code>\swdefault</code> 4, 11, 14, 340, 343, 623, 626	
seq commands:		sys commands:	
<code>\seq_if_empty:NTF</code>	169	<code>\sys_if_engine luatex:TF</code>	3
<code>\seq_new:N</code>	49	<code>\sys_if_engine xetex:TF</code>	10
<code>\seq_put_right:Nn</code>	157, 172	T	
<code>\setboldmathrm</code>	28, 125, 25, 70, 96	T_EX and L^AT_EX 2_ε commands:	
<code>\setfontface</code>	65	<code>\@</code>	31, 64
<code>\setfontfamily</code>	49	<code>\@filelist</code>	50
<code>\SetKeys</code>	36	<code>\@ifpackageloaded</code> 1, 6, 13, 14, 15, 96,	
<code>\setmainfont</code>	24, 28, 123, 6, 24	100, 101, 102, 103, 104, 105, 106, 107,	
<code>\SetMathAlphabet</code>		108, 112, 113, 114, 118, 119, 120, 121,	
... 77, 78, 79, 80, 84, 87, 88, 89, 91, 92		122, 123, 124, 125, 126, 127, 128, 129, 130	
<code>\setmathrm</code>	125, 21, 64, 95	<code>\@nomath</code>	53
<code>\setmathsf</code>	29, 76, 97	<code>\@onlypreamble</code>	95, 96, 97, 98
<code>\setmathtt</code>	33, 82, 98	<code>\@rmfamilyhook</code>	7, 9
<code>\setmonofont</code>	16, 51	<code>\@sffamilyhook</code>	10
<code>\setromanfont</code>	37	<code>\@tempa</code>	29, 30
<code>\setsansfont</code>	11, 38	<code>\@ttfamilyhook</code>	11
<code>\SetSymbolFont</code>	17, 75, 81	<code>\add@unicode@accent</code>	11
		<code>\color@</code>	469, 493
		<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, 200, 203, 204	<code>\tl_if_single:nTF</code>	129, 447
<code>\f@family</code>	3, 3, 28, 41, 44, 47, 48, 93, 95, 123, 138, 154, 155, 200, 203, 204	<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, 160, 164, 165, 166, 167, 168, 169, 170, 171
<code>\f@series</code>	28, 45, 56, 59, 200, 203, 204	<code>\tl_put_left:Nn</code>	46
<code>\f@shape</code>	24, 29, 194	<code>\tl_put_right:Nn</code>	
<code>\f@size</code>	39, 115, 160, 167, 172, 225, 423, 424, 446, 557	9, 10, 11, 137, 431, 441, 453
<code>\reset@font</code>	74	<code>\tl_remove_all:Nn</code>	100, 108, 266
<code>\two@digits</code>	244, 256, 257	<code>\tl_remove_once:Nn</code>	66
tex commands:		<code>\tl_replace_all:Nnn</code>	14, 16, 18, 92, 354
<code>\tex_font:D</code>	59	<code>\tl_set:Nn</code>	21, 22, 22, 23, 25, 26, 28, 28, 34, 38, 39, 42, 45, 46, 47, 49, 53, 55, 60, 67, 81, 86, 99, 104, 105, 112, 115, 116, 117, 123, 124, 128, 129, 130, 146, 149, 150, 151, 155, 161, 164, 165, 168, 172, 173, 197, 225, 226, 259, 263, 265, 275, 281, 284, 293, 307, 308, 322, 323, 327, 328, 342, 342, 343, 350, 353, 358, 359, 364, 373, 386, 388, 392, 394, 398, 398, 400, 414, 414, 419, 424, 448, 449, 475, 490, 499, 514, 520, 523, 532, 542, 546, 581, 598, 639, 649, 661, 740
<code>\tex_hyphenchar:D</code>	54	<code>\tl_set_eq:NN</code>	31, 32, 34, 44, 45, 47, 48, 49, 50, 51, 52, 53, 55, 56, 57, 58, 60, 147, 158, 171, 177, 194, 306, 356, 459, 476, 641, 649, 663, 671, 749, 750, 751
<code>\tex_iffontchar:D</code>	46	<code>\tl_tail:n</code>	706, 720
<code>\textsc</code>	36	<code>\tl_to_str:N</code>	25
<code>\textsf</code>	32	<code>\tl_to_str:n</code>	78, 187
<code>\Theta</code>	54	<code>\tl_trim_spaces:N</code>	56
<code>\tilde</code>	21	<code>\tl_trim_spaces:n</code>	15, 17
<code>\title</code>	31	<code>\tl_use:N</code>	29, 549
tl commands:		<code>\tmpa</code>	27, 28
<code>\c_empty_tl</code> 64, 764, 765, 783, 784, 795, 796		token commands:	
<code>\tl_build_begin:N</code>	457, 458	<code>\token_to_str:N</code>	78, 469, 493
<code>\tl_build_end:N</code>	463, 464	<code>\ttdefault</code>	47, 57, 59, 101, 131
<code>\tl_build_put_right:Nn</code>	526	<code>\ttfamily</code>	7
<code>\tl_clear:N</code>		<code>\typeout</code>	3, 5, 12, 13, 14, 26, 27, 31, 32, 33, 43, 46, 55, 59, 62, 64, 64, 65, 75, 88, 91, 96, 98, 117, 117, 121, 125, 126, 138, 141, 149, 149, 155, 157, 157, 163, 182, 184, 185, 194, 196, 206, 208, 213, 213, 220, 221, 229, 231, 233, 239, 239, 246, 254, 255, 262,
.	23, 24, 44, 128, 136, 297, 307, 353, 475, 732, 733, 734, 735, 748, 752, 753		
<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>	474, 477, 498, 501, 769, 788		
<code>\tl_gclear:N</code>	268, 736, 737, 745, 747		
<code>\tl_gput_right:Nn</code>	614, 693		
<code>\tl_gremove_all:Nn</code>	702		
<code>\tl_gset:Nn</code>	44, 65, 93, 99, 100, 101, 129, 130, 131, 156, 171, 286, 289, 311, 325, 346, 375, 390, 400, 607, 691		
<code>\tl_gset_eq:NN</code>	157, 170, 261, 272, 738		
<code>\tl_if_empty:NTF</code>	29, 46, 50, 82, 96, 211, 243, 257, 258, 282, 327, 365, 386, 394, 398, 411, 480, 493, 521, 540, 589, 644, 654, 655, 662, 663, 664, 666, 713		
<code>\tl_if_empty:nTF</code> 14, 18, 29, 69, 88, 89, 90, 137, 141, 161, 200, 361, 390, 409, 642			
<code>\tl_if_empty_p:N</code>	198		
<code>\tl_if_empty_p:n</code>	87, 92, 142, 190		
<code>\tl_if_eq:NNTF</code>	202, 515, 527		
<code>\tl_if_eq:nnTF</code>	91, 175		
<code>\tl_if_exist:NTF</code>	146, 548		
<code>\tl_if_exist_p:N</code>	24		
<code>\tl_if_in:NnTF</code>	50, 64, 303		
<code>\tl_if_in:nnTF</code>	80, 186		

263, 279, 280, 300, 302, 305, 306, 317, 335, 341, 352, 373, 381, 382, 384, 388, 392, 396, 439, 456, 481, 486, 497, 501, 516, 519, 554, 688, 692, 698, 701, 728, 776	
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>	117, 126
<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>	103, 109
	<code>\use:n</code>
	105, 158, 191, 250, 478
	<code>\use_i:nnn</code>
	312
	<code>\use_ii:nnn</code>
	312
	<code>\use_iii:nnn</code>
	298
	<code>\use_none:nn</code> ...
	88, 89, 90, 91, 92, 93, 94
	<code>\UTFencname</code>
	49, 87
X	
<code>\XeLaTeX</code>	33
<code>\XeTeXcountvariations</code>	387
<code>\XeTeXfeaturename</code>	28
<code>\XeTeXfonttype</code>	380
<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