

The fontspec package

Font selection for X_YLaTeX and LuaLaTeX

WILL ROBERTSON

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

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

2022/01/15 v2.8a

Contents

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

1.3	Generic functions	19
1.4	expl3 variants	20
VI	fontspec-code-fontload.dtx	21
1	expl3 interface for primitive font loading	21
VII	fontspec-code-interfaces.dtx	23
1	User commands	23
VIII	fontspec-code-user.dtx	27
1	User command internals	27
1.1	Font selection	27
1.2	Font feature selection	30
1.3	Defining new font features	32
1.4	High level conditionals	34
1.5	\oldstylenums and \liningnums	35
IX	fontspec-code-api.dtx	36
1	Programmer's interface	36
X	fontspec-code-internal.dtx	42
1	Internals	42
1.1	The main function for setting fonts	42
1.2	Setting font shapes in a family	49
1.3	Initialisation	60
1.4	Miscellaneous	60
XI	fontspec-code-opentype.dtx	62
1	OpenType definitions code	62
1.1	Adding features when loading fonts	63
1.2	OpenType feature information	67
XII	fontspec-code-graphite.dtx	70
1	Graphite/AAT code	70

XIII	fontspec-code-keyval.dtx	72
1	Font loading (keyval) definitions	72
1.1	Pre-pre-parsing stages	72
1.2	Pre-parsed features	74
1.3	Font faces	75
1.4	General font-independent features	78
XIV	fontspec-code-feat-opentype.dtx	88
1	OpenType feature definitions	88
2	Regular key=val / tag definitions	88
2.1	Ligatures	88
2.2	Letters	88
2.3	Numbers	89
2.4	Vertical position	89
2.5	Contextuals	89
2.6	Diacritics	90
2.7	Kerning	90
2.8	Fractions	90
2.9	Style	91
2.10	CJK shape	91
2.11	Character width	92
2.12	Vertical	92
3	OpenType features that need numbering	92
3.1	Alternate	92
3.2	Variant / StylisticSet	93
3.3	CharacterVariant	93
3.4	Annotation	94
3.5	Ornament	94
4	Script and Language	94
4.1	Script	94
4.2	Language	95
5	Backwards compatibility	96
XV	fontspec-code-scripts.dtx	97
1	Font script definitions	97
XVI	fontspec-code-lang.dtx	101
1	Font language definitions	101

XVII	fontspec-code-feat-aat.dtx	109
1	AAT feature definitions	109
1.1	Ligatures	109
1.2	Letters	109
1.3	Numbers	110
1.4	Contextuals	110
1.5	Diacritics	110
1.6	Vertical position	110
1.7	Fractions	110
1.8	Alternate	110
1.9	Variant / StylisticSet	111
1.10	Style	111
1.11	CJK shape	111
1.12	Character width	112
1.13	Annotation	112
XVIII	fontspec-code-enc.dtx	113
1	Extended font encodings	113
XIX	fontspec-code-math.dtx	116
1	Selecting maths fonts	116
XX	fontspec-code-closing.dtx	121
1	Closing code	121
1.1	Finishing up	121
XXI	fontspec-code-xfss.dtx	122
1	Changes to the NFSS	122
1.1	Italic small caps and so on	122
1.2	Strong emphasis	122
	Index	125

File I

fontspec.dtx

1 Package declaration

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

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

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

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

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

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

```

40 }
41 </dtx>

```

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

```

42 <fontspec>\RequirePackage{xparse}
43 <fontspec & load>\ProvidesExplPackage{fontspec}%
44 <fontspec & XE>\ProvidesExplPackage{fontspec-xetex}%
45 <fontspec & LU>\ProvidesExplPackage{fontspec-luatex}%
46 *dtx)
47 \RequirePackage{xparse}
48 \ProvidesExplFile{fontspec.dtx}
49 </dtx>
50 *fontspec)
51 {2022/01/15}{2.8a}{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        = "2.8a",
62 (lua)  date           = "2022/01/15",
63 (lua)  description    = "Font selection for XeLaTeX and LuaLaTeX",
64 (lua)  author         = "Khaled Hosny, Philipp Gesang, Will Robertson",
65 (lua)  copyright      = "Khaled Hosny, Philipp Gesang, Will Robertson",
66 (lua)  license        = "LPPL v1.3c"
67 (lua)}

```

File II

fontspec-code-load.dtx

1 The fontspec.sty loading file

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

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

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

```
2 <*load>
```

Lua[®]TeX

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

X₃TeX

```
10 \sys_if_engine_xetex:T
11 {
12   \RequirePackageWithOptions{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 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 \bool_new:N \g_@@_cfg_bool
```

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

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

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

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

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

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

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

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

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

```
27 \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!

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

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

Counters

```
29 \int_new:N \l_@@_script_int
30 \int_new:N \l_@@_language_int
31 \int_new:N \l_@@_strnum_int
32 \int_new:N \l_@@_tmp_int
33 \int_new:N \l_@@_tmpa_int
34 \int_new:N \l_@@_tmpb_int
35 \int_new:N \l_@@_tmpc_int
36 \int_new:N \l_@@_em_int
37 \int_new:N \l_@@_emdef_int
38 \int_new:N \l_@@_strong_int
39 \int_new:N \l_@@_strongdef_int
```

Floats

```
40 \fp_new:N \l_@@_tmpa_fp
41 \fp_new:N \l_@@_tmpb_fp
```

Dimensions

```
42 \dim_new:N \l_@@_tmpa_dim
43 \dim_new:N \l_@@_tmpb_dim
44 \dim_new:N \l_@@_tmpc_dim
```

Sequences

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

Comma-lists

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

```

61 \clist_new:N \l_@@_fontfeat_up_clist
62 \clist_new:N \l_@@_fontfeat_bf_clist
63 \clist_new:N \l_@@_fontfeat_it_clist
64 \clist_new:N \l_@@_fontfeat_bfit_clist
65 \clist_new:N \l_@@_fontfeat_sl_clist
66 \clist_new:N \l_@@_fontfeat_bfsl_clist
67 \clist_new:N \l_@@_fontfeat_sw_clist
68 \clist_new:N \l_@@_fontfeat_bfsw_clist
69 \clist_new:N \l_@@_fontfeat_sc_clist

```

Property lists

```

70 \prop_new:N \g_@@_fontopts_prop
71 \prop_new:N \l_@@_nfss_prop
72 \prop_new:N \l_@@_nfssfont_prop
73 \prop_new:N \g_@@_OT_features_prop
74 \prop_new:N \g_@@_all_opentype_feature_names_prop
75 \prop_new:N \g_@@_em_prop
76 \prop_new:N \g_@@_strong_prop
77 \prop_new:N \g_@@_fontid_family_prop
78 \prop_new:N \g_@@_family_int_prop

```

Token lists

Visible (perhaps?)

```

79 \tl_new:N \l_fontspeg_family_tl
80 \tl_new:N \g_fontspeg_encoding_tl
81 \tl_new:N \l_fontspeg_fontname_tl

```

2e interactions

```

82 \tl_clear_new:N \UTFencname
83 \tl_clear_new:N \cyrillicencoding
84 \tl_clear_new:N \latinencoding

```

Renderer/shaper

```

85 \tl_new:N \l_@@_renderer_tl
86 \tl_new:N \l_@@_mode_tl
87 \tl_new:N \l_@@_shaper_tl

88 \tl_new:N \g_@@_defined_shapes_tl
89 \tl_new:N \g_@@_single_feat_tl
90 \tl_new:N \l_@@_basename_tl
91 \tl_new:N \g_@@_curr_series_tl
92 \tl_new:N \l_@@_curr_fontname_tl
93 \tl_new:N \l_@@_curr_bfname_tl
94 \tl_new:N \l_@@_ext_filename_tl
95 \tl_new:N \l_@@_extension_tl
96 \tl_new:N \l_@@_font_path_tl
97 \tl_new:N \l_@@_fontid_tl
98 \tl_new:N \l_@@_fontname_tl

```

```

99 \tl_new:N \l_@@_options_tl
100 \tl_new:N \l_@@_saved_fontname_tl
101 \tl_new:N \l_@@_prev_unicode_name_tl

102 \tl_new:N \g_@@_nfss_enc_tl
103 \tl_new:N \g_@@_nfss_family_tl
104 \tl_new:N \l_@@_nfss_sc_tl
105 \tl_new:N \l_@@_nfss_tl
106 \tl_new:N \l_@@_nfss_fam_tl

107 \tl_new:N \l_@@_size_tl
108 \tl_new:N \l_@@_sizedfont_tl
109 \tl_new:N \l_@@_this_font_tl
110 \tl_new:N \l_@@_ttc_index_tl
111 \tl_new:N \l_@@_smcp_shape_tl

```

EM and STRONG

```

112 \tl_new:N \l_@@_emshape_query_tl
113 \tl_new:N \l_@@_em_switch_tl
114 \tl_new:N \l_@@_strong_switch_tl

```

Scratch variables

```

115 \tl_new:N \l_@@_tmp_tl
116 \tl_new:N \l_@@_tmpa_tl
117 \tl_new:N \l_@@_tmpb_tl
118 \tl_new:N \l_@@_em_tmp_tl
119 \tl_new:N \l_@@_strong_tmp_tl

```

Maths fonts

```

120 \tl_new:N \g_@@_mathrm_tl
121 \tl_new:N \g_@@_bfmathrm_tl
122 \tl_new:N \g_@@_mathsf_tl
123 \tl_new:N \g_@@_mathtt_tl

```

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

```

124 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
125 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
126 \tl_gset:Nn \g_@@_mathtt_tl {\ttdefault}

127 \tl_new:N \l_@@_family_label_tl
128 \tl_new:N \l_@@_fake_slant_tl
129 \tl_new:N \l_@@_fake_embolden_tl

```

Internal font names

```

130 \tl_new:N \l_@@_fontname_up_tl
131 \tl_new:N \l_@@_fontname_bf_tl
132 \tl_new:N \l_@@_fontname_it_tl
133 \tl_new:N \l_@@_fontname_bfit_tl
134 \tl_new:N \l_@@_fontname_sl_tl
135 \tl_new:N \l_@@_fontname_bfsl_tl
136 \tl_new:N \l_@@_fontname_sw_tl

```

```

137 \tl_new:N \l_@@_fontname_bfsw_tl
138 \tl_new:N \l_@@_fontname_sc_tl

```

Script and Language

```

139 \tl_new:N \l_@@_script_tl
140 \tl_new:N \l_@@_script_name_tl
141 \tl_set:Nn \l_@@_script_name_tl {CustomDefault}

142 \tl_new:N \l_@@_lang_tl
143 \tl_new:N \l_@@_lang_name_tl
144 \tl_set:Nn \l_@@_lang_name_tl {Default}

```

Generic font features

```

145 \tl_new:N \l_@@_scale_tl
146 \tl_new:N \l_@@_hyphenchar_tl
147 \tl_new:N \l_@@_hexcol_tl
148 \tl_new:N \l_@@_opacity_tl
149 \tl_new:N \l_@@_optical_size_tl
150 \tl_new:N \l_@@_mapping_tl
151 \tl_new:N \l_@@_punctspace_adjust_tl
152 \tl_new:N \l_@@_wordspace_adjust_tl
153 \tl_new:N \l_@@_postadjust_tl

154 \tl_const:Nn \c_@@_hexcol_tl {000000}
155 \tl_const:Nn \c_@@_opacity_tl {FF~}
156 \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.

```

157 \tl_new:N \g_@@_rawfeatures_sclist
158 \tl_new:N \l_@@_pre_feat_sclist

```

Font families

```

159 \tl_new:N \l_@@_rmfamily_family_tl
160 \tl_new:N \l_@@_sffamily_family_tl
161 \tl_new:N \l_@@_ttfamily_family_tl
162 \tl_new:N \l_@@_rmfamily_encoding_tl
163 \tl_new:N \l_@@_sffamily_encoding_tl
164 \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 \@@_warning:n { \msg_warning:nn {fontspec} }
5 \cs_new:Npn \@@_warning:nx { \msg_warning:nnx {fontspec} }
6 \cs_new:Npn \@@_warning:nxx { \msg_warning:nnxx {fontspec} }
7 \cs_new:Npn \@@_info:n { \msg_info:nn {fontspec} }
8 \cs_new:Npn \@@_info:nx { \msg_info:nnx {fontspec} }
9 \cs_new:Npn \@@_info:nxx { \msg_info:nnxx {fontspec} }
10 \cs_new:Npn \@@_trace:n { \msg_trace:nn {fontspec} }
```

Allow messages to be written with spaces acting as normal:

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

1.1 Errors

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

```

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

```

1.2 Warnings

```

84 \@@_msg_new:nn {tu-clash}
85 {
86 I have found the tuenc.def encoding definition file but the TU encoding is not
87 defined by the LaTeX2e kernel; attempting to correct but you really should update

```

```

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

```

```

139 Language '#1' not available
140 for font '\l_fontspec_fontname_tl'
141 with script '\l_@@_script_name_tl'.
142 }
143 \@@_msg_new:nn {only-xetex-feature}
144 {
145   Ignored XeTeX-only feature: '#1'.
146 }
147 \@@_msg_new:nn {only-luatex-feature}
148 {
149   Ignored LuaTeX-only feature: '#1'.
150 }
151 \@@_msg_new:nn {unknown-renderer}
152 {
153   Renderer '#1' unknown. Assuming Harfbuzz with 'shaper=#1'.
154   Please raise a fontspec issue to add this shaper to the interface.
155 }
156 \@@_msg_new:nn {no-mapping}
157 {
158   Input mapping not supported in LuaTeX.
159 }
160 \@@_msg_new:nn {no-mapping-ligtex}
161 {
162   Input mapping not supported in LuaTeX.\\
163   Use "Ligatures=TeX" instead of "Mapping=tex-text".
164 }
165 \@@_msg_new:nn {cm-default-obsolete}
166 {
167   The "cm-default" package option is obsolete.
168 }
169 \@@_msg_new:nn {font-index-needs-ttc}
170 {
171   The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
172   Feature ignored.
173 }
174 \@@_msg_new:nn {feat-cannot-remove}
175 {
176   The "#1" feature cannot be deactivated. Request ignored.
177 }

```

1.3 Info messages

```

178 \@@_msg_new:nn {defining-font}
179 {
180   Font family '\g_@@_nfss_family_tl' created for font '#2'
181   with options [\l_@@_all_features_clist].\\
182   \\
183   This font family consists of the following NFSS series/shapes:\\
184   \g_@@_defined_shapes_tl
185 }
186 \@@_msg_new:nn {no-font-shape}
187 {

```



```

188   Could not resolve font "#1" (it probably doesn't exist).
189   }
190   \@@_msg_new:nn {set-scale}
191   {
192     \l_fontspec_fontname_tl\space scale = \l_@@_scale_tl.
193   }
194   \@@_msg_new:nn {setup-math}
195   {
196     Adjusting the maths setup (use [no-math] to avoid this).
197   }
198   \@@_msg_new:nn {no-script}
199   {
200     Font "#1" does not contain requested Script "#2".
201   }
202   \@@_msg_new:nn {opa-twice}
203   {
204     Opacity set twice, in both Colour and Opacity.\
205     Using specification "Opacity=#1".
206   }
207   \@@_msg_new:nn {opa-twice-col}
208   {
209     Opacity set twice, in both Opacity and Colour.\
210     Using an opacity specification in hex of "#1/FF".
211   }
212   \@@_msg_new:nn {bad-colour}
213   {
214     Bad colour declaration "#1".
215     Colour must be one of:\
216     * a named xcolor colour\
217     * a six-digit hex colour RRGGBB\
218     * an eight-digit hex colour RRGGBBTT with opacity
219   }
220   Reset 'space' behaviour:
221   \char_set_catcode_ignore:n {32}

```

File V

fontspec-code-opening.dtx

1 Opening code

1.1 Package options

```
1 \DeclareOption{cm-default}
2 {
3   \@@_warning:n {cm-default-obsolete}
4 }
5
6 \DeclareOption {math}      { \bool_gset_true:N \g_@@_math_bool }
7 \DeclareOption {no-math}  { \bool_gset_false:N \g_@@_math_bool }
8 \DeclareOption {config}   { \bool_gset_true:N \g_@@_cfg_bool }
9 \DeclareOption {no-config}{ \bool_gset_false:N \g_@@_cfg_bool }
10 \DeclareOption {euenc}    { \bool_gset_true:N \g_@@_euenc_bool }
11 \DeclareOption {tuenc}    { \bool_gset_false:N \g_@@_euenc_bool }
12
13 \DeclareOption {quiet}
14 {
15   \msg_redirect_module:nnn { fontspec } { warning } { info }
16   \msg_redirect_module:nnn { fontspec } { info } { none }
17 }
18
19 \DeclareOption{silent}
20 {
21   \msg_redirect_module:nnn { fontspec } { warning } { none }
22   \msg_redirect_module:nnn { fontspec } { info } { none }
23 }
24
25 \ExecuteOptions{config,math,tuenc}
26 \ProcessOptions*
```

1.2 Encodings

Soon to be the default, with a just-in-case check:

```
27 \bool_if:NF \g_@@_euenc_bool
28 {
29   \file_if_exist:nTF {tuenc.def}
30   {
31     \cs_if_exist:cF {T@TU}
32     {
33       \@@_warning:n {tu-clash}
34       \DeclareFontEncoding{TU}{}{}
35       \DeclareFontSubstitution{TU}{lmr}{m}{n}
36     }
37   }
38   {
39     \@@_warning:n {tu-missing}
40     \bool_gset_true:N \g_@@_euenc_bool
41   }
42 }
```

```

39 \bool_if:NTF \g_@@_euenc_bool
40 {
41 <XE> \tl_gset:Nn \g_fontspec_encoding_tl {EU1}
42 <LU> \tl_gset:Nn \g_fontspec_encoding_tl {EU2}
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.

```

58 \bool_if:NT \g_@@_euenc_bool
59 {
60 <LU> \cs_set_eq:NN \fontspec_tmp: \XeTeXpicfile
61 <LU> \cs_set:Npn \XeTeXpicfile {}
62 \RequirePackage{xunicode}
63 <LU> \cs_set_eq:NN \XeTeXpicfile \fontspec_tmp:
64 }

```

1.3 Generic functions

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

```

\FontspecSetCheckBoolFalse 65 \cs_new:Npn \FontspecSetCheckBoolTrue { \bool_set_true:N \l_@@_check_bool }
66 \cs_new:Npn \FontspecSetCheckBoolFalse { \bool_set_false:N \l_@@_check_bool }

```

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

`\@@_keys_set_known:nnN`

```

67 \cs_new:Nn \@@_keys_set_known:nnN
68 {
69 <debug> \typeout{::: Keys~set:~{#1}~{#2} }
70 \keys_set_known:nnN {#1} {#2} #3
71 <debug> \typeout{::: Leftover:~{#3} }
72 }
73 \cs_generate_variant:Nn \@@_keys_set_known:nnN {nx}

```

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

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

```
74 \cs_new:Nn \@@_int_mult_truncate:Nn
75 {
76   \int_set:Nn #1 { \_dim_eval:w #2 #1 \_dim_eval_end: }
77 }
```

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

```
\@@_lua_function:ne
\@@_lua_function:nee 78 {*LU}
\@@_lua_function:neee 79 \cs_set:Npn \@@_lua_function:ne #1#2 { \lua_now:e { fontspec.#1 ("#2")
\@@_lua_function:neeee 80 \cs_set:Npn \@@_lua_function:nee #1#2#3 { \lua_now:e { fontspec.#1 ("#2", "#3")
81 \cs_set:Npn \@@_lua_function:neee #1#2#3#4 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4")
82 \cs_set:Npn \@@_lua_function:neeee #1#2#3#4#5 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4", "#5")
83 </LU>
```

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

1.4 expl3 variants

```
84 \cs_generate_variant:Nn \int_set:Nn {Nv}
85 \cs_generate_variant:Nn \keys_set:nn {nx}
86 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
87 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
88 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
89 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
90 \cs_generate_variant:Nn \prop_gput:Nnn {NxN}
91 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
92 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
93 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
94 \cs_generate_variant:Nn \tl_if_empty_p:n {e}
95 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
96 \cs_generate_variant:Nn \tl_if_empty:nF {x}
97 \cs_generate_variant:Nn \tl_if_empty:nF {f}
98 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
99 \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 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 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_set_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 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_set_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_set_conditional:Nnn \@@_primitive_font_gset:Nnn {TF,T,F}
27 {
28   \@@_primitive_font_gset:Nnn #1 {#2} {#3}
29   \@@_primitive_font_if_null:NTF #1 {\prg_return_false:} {\prg_return_true:}
30 }
31 \cs_set:Npn \@@_primitive_font_set:Omm { \exp_last_unbraced:No \@@_primitive_font_set:Nnn }
```

```

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

```

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

\@@_primitive_font_if_exist:nTF

```

35 \prg_set_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 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 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 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 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 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 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 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 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 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 definition for \oldstylenums. This function is documented on page ??.)
```

`\liningnums`

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

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

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

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

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

```

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

```

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

`\setsansfont` Same as above.

```

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

```

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

`\setmonofont` Same as above.

```

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

```

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

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

```

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

```

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

`\setboldmathrm`

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

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

`\setmathsf`

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

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

`\setmathtt`

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

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

Hooks:

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

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

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

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

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

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

```

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

```

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

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

```

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

```

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

1.2 Font feature selection

`\@@_set_default_features:nn`

```

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

```

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

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

```

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

```

```

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

```

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

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

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

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

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

```

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

```

```

168     \ignorespaces
169   }

```

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

1.3 Defining new font features

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

```

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

```

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

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

```

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

```

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

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

```

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

```



```

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

```

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

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

```

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

```

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

\aliasfontfeatureoption

```

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

```

```

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

```

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

\@@_main_DeclareFontExtensions:n

```

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

```

Defaults:

```

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

```

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

1.4 High level conditionals

\IfFontFeatureActiveTF

```

259 \cs_new:Nn \@@_main_IfFontFeatureActiveTF:nnn
260 {
261 (debug) \typeout{^^J::::::::::::::::::::::::::::::::::::::::::::::::::}
262 (debug) \typeout{:IfFontFeatureActiveTF \exp_not:n{#{#1}{#2}{#3}}}
263     \@@_if_font_feature:nTF {#1} {#2} {#3}
264 }

```

```

265 \prg_new_conditional:Nnn \@@_if_font_feature:n {TF}
266 {
267     \tl_gclear:N \g_@@_single_feat_tl
268     \group_begin:
269         \@@_font_suppress_not_found_error:
270         \@@_init:
271         \bool_set_true:N \l_@@_ot_bool
272         \bool_set_true:N \l_@@_never_check_bool
273         \bool_set_false:N \l_@@_firsttime_bool
274         \clist_clear:N \l_@@_fontfeat_clist
275         \@@_get_features:n {#1}
276     \group_end:
277
278 (debug) \typeout{::: > \exp_not:N\g_@@_rawfeatures_sclist->~{\g_@@_rawfeatures_sclist}}
279 (debug) \typeout{::: > \exp_not:N\g_@@_single_feat_tl->~{\g_@@_single_feat_tl}}
280
281     \tl_if_empty:NTF \g_@@_single_feat_tl { \prg_return_false: }

```

```

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

```

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

1.5 \oldstylenums and \liningnums

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

```

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

```

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

File IX

fontspec-code-api.dtx

1 Programmer's interface

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

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

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

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

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

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

144     { \prg_return_false: }
145   }

```

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

```

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

```

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

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

```

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

```

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

```

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

```

```

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

```

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

```

\fontspec_font_if_exist:n

```

```

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

```



```

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

```

```

180 \cs_set_eq:NN \IfFontExistsTF \fontspec_font_if_exist:nTF

```

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

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

```

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

```

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

`\fontspec_if_small_caps:TF`

```

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

```

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

File X

fontspec-code-internal.dtx

1 Internals

1.1 The main function for setting fonts

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

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

```

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

```

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

\@@_extract_all_features:

```

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

```

```

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

```

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

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

Perform the (multi-step) feature parsing process.

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

```

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

```

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

```

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

```

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

```

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

```

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

\@@_load_font:

```

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

```

```

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

```

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

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

#2 : Extension

#3 : TTC Index

#4 : Renderer

#5 : Optical size

#6 : Font features

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

```

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

```

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

```

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

```

(End 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'. X₃TEX's syntax is followed since `luaotfload` provides compatibility.

```

167 \cs_new:Nn \@@_font_is_name:
168   {
169     \cs_set_eq:NN \@@_fontname_wrap:n \use:n
170   }
171 \cs_new:Nn \@@_font_is_file:
172   {
173     \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
174   }

```

(End 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.

```

175 \cs_new:Nn \@@_set_scriptlang:
176   {
177     <debug> \typeout{:: _set_scriptlang:}
178     \bool_if:NT \l_@@_firsttime_bool
179       {
180         \tl_if_empty:NF \l_@@_script_name_tl
181           {
182             <debug> \typeout{::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
183             \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
184             \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
185           }
186       }
187   }

```

(End 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.

```

188 \cs_new:Nn \@@_get_features:n
189   {
190     <debug> \typeout{:: @@_get_features:Nn { \exp_not:n {#1} } }
191     \@@_init_fontface:
192     \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#1}
193     \l_@@_keys_leftover_clist
194     \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
195     <*XE>
196     \bool_if:NTF \l_@@_ot_bool
197       {
198         <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist}
199         \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
200       }
201       {
202         <debug> \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features:~"\l_@@_keys_leftover_c

```

```

203     \bool_if:nT { \l_@@_atsui_bool || \l_@@_graphite_bool }
204     { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
205   }
206 </XE>
207 <*LU>
208 (debug) \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
209     \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
210 </LU>
211
212   \tl_if_empty:NF \l_@@_mapping_tl
213     { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
214
215   \str_if_eq:eeF { \l_@@_hexcol_tl \l_@@_opacity_tl }
216     { \c_@@_hexcol_tl \c_@@_opacity_tl }
217     { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
218   }

```

(End 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.

```

219 \prg_new_conditional:Nnn \@@_save_family_needed:n { TF }
220 {
221
222 (debug) \typeout{save~ family:~ #1}
223 (debug) \typeout{== fontid_tl: "\l_@@_fontid_tl".}
224
225   \tl_if_empty:NTF \l_@@_nfss_fam_tl
226     {
227     \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
228       {
229       \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
230       \prg_return_false:
231       }
232     {
233     \tl_set:Nx \l_@@_tmp_tl {#1}
234     \tl_remove_all:Nn \l_@@_tmp_tl { ~ }
235     \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl
236     \prg_return_true:
237     }
238   }
239   {
240   \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
241   \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl_prop }
242   \prg_return_true:
243   }
244 }

```



```

245 \cs_new:Nn \@@_save_fontid_family:nn
246 {
247   \prop_get:NnNTF \g_@@_family_int_prop {#2} \l_@@_tmp_tl
248   {
249     \tl_set:Nx \l_@@_tmp_tl
250     { \int_eval:n { \l_@@_tmp_tl + 1 } }
251   }
252   { \tl_set:Nn \l_@@_tmp_tl { 0 } }
253   \prop_gput:NnV \g_@@_family_int_prop {#2} \l_@@_tmp_tl
254   \tl_gset:Nx \g_@@_nfss_family_tl { #2 ( \l_@@_tmp_tl ) }
255   \prop_gput:NnV \g_@@_fontid_family_prop {#1} \g_@@_nfss_family_tl
256 }
257 \cs_generate_variant:Nn \@@_save_fontid_family:nn { VV }

```

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

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

```

258 \cs_new:Nn \@@_save_family:nn
259 {
260   \@@_save_fontinfo:n {#2}
261   \@@_find_autofonts:
262   \DeclareFontFamily{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{
263     \@@_set_faces:
264     \@@_info:nxx {defining-font} {#1} {#2}
265   }

```

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

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

```

266 \cs_new:Nn \@@_save_fontinfo:n
267 {
268   \prop_new:c {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop}
269   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
270   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options} { \l_@@_all_features }
271   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
272   {
273     \@@_construct_font_call:nn {\l_fontspeg_fontname_tl}
274     { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
275   }
276   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
277   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num} \l_@@_language_int
278   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_@@_script_tl
279   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_@@_lang_tl
280 }

```

(End 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 `\bdfdefault` 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:`

```

281 \cs_new:Nn \@@_find_autofonts:
282 {
283   \bool_if:NF {\l_@@_noit_bool || \l_@@_nobf_bool}
284   {
285     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
286     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
287     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspeg_fontname_tl} {/BI}
288   }
289
290   \bool_if:NF \l_@@_nobf_bool
291   {
292     \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspeg_fontname_tl} {/B}
293   }
294
295   \bool_if:NF \l_@@_noit_bool
296   {
297     \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspeg_fontname_tl} {/I}
298   }
299
300   \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
301 }

```

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

`\@@_set_faces:`

```

302 \cs_new:Nn \@@_set_faces:
303 {
304   \@@_add_nfssfont:nnnn \mddefault \shapedefault \l_fontspeg_fontname_tl \l_@@_fontfeat_up_c
305   \@@_add_nfssfont:nnnn \bfdefault \shapedefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_c
306   \@@_add_nfssfont:nnnn \mddefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_c
307   \@@_add_nfssfont:nnnn \mddefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_c
308   \@@_add_nfssfont:nnnn \mddefault \swdefault \l_@@_fontname_sw_tl \l_@@_fontfeat_sw_c
309   \@@_add_nfssfont:nnnn \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit
310   \@@_add_nfssfont:nnnn \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl
311   \@@_add_nfssfont:nnnn \bfdefault \swdefault \l_@@_fontname_bfsw_tl \l_@@_fontfeat_bfsw
312   \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nnnnn ##2 }
313 }
314 \cs_new:Nn \@@_set_faces_aux:nnnnn
315 {
316   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
317   \@@_make_font_shapes:Nnnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
318 }

```

(End 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”).

```

319 \cs_new:Nn \fontspec_complete_fontname:Nn
320 {
321   \tl_set:Nx #1 {#2}
322   \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
323   \tl_remove_all:Nn #1 {~}
324 }

```

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

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

```

325 \cs_new:Nn \@@_add_nfssfont:nmnn
326 {
327   \tl_set:Nx \l_@@_this_font_tl {#3}
328
329   \tl_if_empty:xTF {#4}
330   { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
331   { \@@_keys_set_known:nxN {fontspec-preparse-nested} {#4} \l_@@_tmp_tl }
332
333   \tl_if_empty:NF \l_@@_this_font_tl
334   {
335     \prop_put:Nxx \l_@@_nfssfont_prop {#1/#2}
336     { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
337   }
338 }

```

(End definition for `\@@_add_nfssfont:nmnn`. 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_fontspec_test_font` is an AAT font or an OpenType font or a font with feature axes (either AAT or Multiple Master), respectively.

```

339 \cs_new:Nn \@@_set_font_type:N
340 {
341   \debug \typeout{:: @@@_set_font_type:}
342   \*XE)
343   \bool_set_false:N \l_@@_tfm_bool
344   \bool_set_false:N \l_@@_atsui_bool
345   \bool_set_false:N \l_@@_ot_bool
346   \bool_set_false:N \l_@@_mm_bool
347   \bool_set_false:N \l_@@_graphite_bool
348   \ifcase\XeTeXfonttype #1
349   \debug \typeout{::: TFM}

```

```

350 \bool_set_true:N \l_@@_tfm_bool
351 \or
352 <debug> \typeout{::: AAT}
353 \bool_set_true:N \l_@@_atsui_bool
354 \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/AAT} }
355 \ifnum\XeTeXcountvariations #1 > 0\relax
356 <debug> \typeout{::: MM}
357 \bool_set_true:N \l_@@_mm_bool
358 \fi
359 \or
360 <debug> \typeout{::: OpenType}
361 \bool_set_true:N \l_@@_ot_bool
362 \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/OT} }
363 \or
364 <debug> \typeout{::: Graphite}
365 \bool_set_true:N \l_@@_graphite_bool
366 \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/GR} }
367 \fi
368 </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:

```

369 <*LU>
370     \bool_set_true:N \l_@@_ot_bool
371 </LU>
372 }

```

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

```

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

```

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

We can't match external fonts in this way (in X_YTeX 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.

```

373 \cs_new:Nn \@@_set_autofont:Nnn
374 {
375     \bool_if:NF \l_@@_external_bool
376     {
377         \tl_if_empty:xF {#2}
378         {
379             \tl_if_empty:NT #1
380             {
381                 \@@_if_autofont:nnTF {#2} {#3}

```

```

382         { \tl_set:Nx #1 {#2#3} }
383         { \@@_info:nx {no-font-shape} {#2#3} }
384     }
385 }
386 }
387 }
388 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
389 {
390     \group_begin:
391     \@@_primitive_font_set:Nnn \l_@@_tmpa_font { \@@_construct_font_call:nn {#1} { \l_@@_pre
392     \@@_primitive_font_set:Nnn \l_@@_tmpb_font { \@@_construct_font_call:nn {#1#2} { \l_@@_pre
393     \str_if_eq:eeTF { \@@_primitive_font_get_name:N \l_@@_tmpa_font } { \@@_primitive_font_get
394     { \group_end: \prg_return_false: }
395     { \group_end: \prg_return_true: }
396 }

```

(End 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.

```

397 \cs_new:Nn \@@_make_font_shapes:Nnnnn
398 {
399     \group_begin:
400     \@@_keys_set_known:nxN {fontspec-prepare-external} { #4 } \l_@@_leftover_clist
401     \@@_load_fontname:Nn \l_fontspeg_fontname_tl {#1}
402     \@@_declare_shape:nxxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
403     \group_end:
404 }
405 \cs_new:Nn \@@_load_fontname:Nn
406 {
407     <debug> \typeout{: : \@@_load_fontname:Nn \exp_not:N #1 (#1) {#2} }
408     \@@_sanitise_fontname:Nn #1 {#2}
409     \@@_load_external_fontoptions:N #1
410     \prop_get:NVNF \g_@@_fontopts_prop #1 \l_@@_fontopts_clist
411     { \clist_clear:N \l_@@_fontopts_clist }
412     \keys_set_groups:nnV {fontspec/fontname} {getfontname} \l_@@_fontopts_clist
413     \@@_primitive_font_set:OnnF \l_@@_fontface_cs_tl
414     { \@@_construct_font_call:nn {#1} { \l_@@_pre_feat_sclist } } { \f@size pt + 2sp }
415     { \@@_error:nx {font-not-found} {#2} }
416 }
417 \keys_define:nn {fontspec/fontname}
418 {
419     Font .tl_set:N = \l_fontspeg_fontname_tl ,
420     Font .groups:n = {getfontname} ,
421 }

```

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

```

\@@_declare_shape:nmmn #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>}}.

```

422 \cs_new:Nn \@@_declare_shape:nmmn
423   {
424   (debug)\typeout{== declare_shape:~{\l_fontspeg_fontname_tl}~{#1}~{#2}}
425     \tl_build_begin:N \l_@@_nfss_tl
426     \tl_build_begin:N \l_@@_nfss_sc_tl
427     \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspeg_fontname_tl
428
429     \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
430
431     \tl_build_end:N \l_@@_nfss_tl
432     \tl_build_end:N \l_@@_nfss_sc_tl
433
434     \@@_declare_shapes_normal:nn {#1} {#2}
435     \@@_declare_shapes_smcaps:nn {#1} {#2}
436     \@@_declare_shape_slanted:nn {#1} {#2}
437     \@@_declare_shapes_bx:nn {#1} {#2}
438     \@@_declare_shape_loginfo:nn {#1} {#2}
439   }
440 \cs_generate_variant:Nn \@@_declare_shape:nmmn {nxxx}

```

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

```

\@@_setup_single_size:nn

```

```

441 \cs_new:Nn \@@_setup_single_size:nn
442   {
443     \tl_clear:N \l_@@_size_tl
444     \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
445
446     \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
447     \l_@@_sizing_leftover_clist
448     \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
449   (debug)\typeout{== size:~\l_@@_size_tl}
450
451     % "normal"
452     \@@_load_fontname:Nn \l_fontspeg_fontname_tl {\l_@@_sizedfont_tl}
453     \@@_setup_nfss:Nmmn \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
454   (debug) \typeout{=== sized~ font:~ \l_@@_sizedfont_tl}
455
456     % small caps
457     \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist
458

```

```

459     \bool_if:NF \l_@@_nosc_bool
460     {
461         \tl_if_empty:NTF \l_@@_fontname_sc_tl
462         {
463             \@@_make_smallcaps:TF
464             {
465 (debug)\typeout{====~Small~ caps~ found.}
466                 \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
467             }
468             {
469 (debug)\typeout{====~Small~ caps~ not~ found.}
470                 \bool_set_true:N \l_@@_nosc_bool
471             }
472         }
473         { \@@_load_fontname:Nn \l_fontspec_fontname_tl {\l_@@_fontname_sc_tl }% local for e
474     }
475
476     \bool_if:NF \l_@@_nosc_bool
477     {
478         \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
479         {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
480     }
481 }

```

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

\@@_setup_nfss:Nnnn

```

482 \cs_new:Nn \@@_setup_nfss:Nnnn
483 {
484 (debug)\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}
485
486     \@@_get_features:n { #2 , #3 , #4 }
487 (debug)\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist}
488
489     \tl_if_empty:NF \l_@@_scale_tl
490     {
491         \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
492     }
493
494     \tl_build_put_right:Nx #1
495     {
496         <\l_@@_size_tl> \l_@@_scale_tl
497         \@@_construct_font_call:nn { \l_fontspec_fontname_tl }
498         { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
499     }
500 }

```

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

\@@_declare_shapes_normal:nn

```

501 \cs_new:Nn \@@_declare_shapes_normal:nn
502 {

```

```

503 \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
504 {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
505 }

```

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

\@@_declare_shapes_smcaps:nn

```

506 \cs_new:Nn \@@_declare_shapes_smcaps:nn
507 {
508   \tl_if_empty:NF \l_@@_nfss_sc_tl
509   {
510     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl} {#1}
511     { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
512   }
513 }
514 \cs_new:Nn \@@_combo_sc_shape:n
515 {
516   \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
517   { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
518   { \scdefault#1 }
519 }

```

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

\@@_DeclareFontShape:nnnnnn

```

520 \cs_new:Nn \@@_DeclareFontShape:nnnnnn
521 {
522   <debug>\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
523   \group_begin:
524   \normalsize
525   \cs_undefine:c {#1/#2/#3/#4/\f@size}
526   \group_end:
527   \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
528   }
529 \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.

\@@_declare_shape_slanted:nn

```

530 \cs_new:Nn \@@_declare_shape_slanted:nn
531 {
532   \bool_if:nT
533   {
534     \str_if_eq_p:ee {#2} {\itdefault} &&
535     !(\str_if_eq_p:ee {\itdefault} {\sldefault})
536   }
537   {
538     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}

```



```

539         {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
540     }
541 }

```

Similar processing for setting up b/bx substitutions.

```

\@@_declare_shapes_bx:nn 542 \cs_new:Nn \@@_declare_shapes_bx:nn
543 {
544     \bool_if:nT
545     {
546         \str_if_eq_p:ee {#1} {\bfdefault} &&
547         !(\str_if_eq_p:ee {\bfdefault} {bx})
548     }
549     {
550         % bx/?
551         \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
552         {bx} {#2}
553         { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
554         { \l_@@_postadjust_tl }
555
556         % bx/sc -> b/sc
557         \tl_if_empty:NF \l_@@_nfss_sc_tl
558         {
559             \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
560             {bx} { \@@_combo_sc_shape:n {#2} }
561             { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
562             { \l_@@_postadjust_tl }
563         }
564
565         % bx/sl -> bx/it
566         \bool_if:nT
567         {
568             \str_if_eq_p:ee {#2} {\itdefault} &&
569             !(\str_if_eq_p:ee {\itdefault} {\sldefault})
570         }
571         {
572             \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
573             {bx} {\sldefault}
574             { <->ssub*\g_@@_nfss_family_tl/bx/\itdefault }
575             { \l_@@_postadjust_tl }
576         }
577     }
578 }
579 }

```

Lastly some informative messaging.

```

\@@_declare_shape_loginfo:nn 580 \cs_new:Nn \@@_declare_shape_loginfo:nn
581 {
582     \tl_gput_right:Nx \g_@@_defined_shapes_tl
583     {
584         \exp_not:n { \ }
585         -- \exp_not:N \str_case:nn {#1/#2}

```

```

586 {
587   {\mddefault/\shapedefault} {'normal'~}
588   {\bfdefault/\shapedefault} {'bold'~}
589   {\mddefault/\itdefault} {'italic'~}
590   {\mddefault/\sldefault} {'slanted'~}
591   {\mddefault/\swdefault} {'swash'~}
592   {\bfdefault/\itdefault} {'bold~ italic'~}
593   {\bfdefault/\sldefault} {'bold~ slanted'~}
594   {\bfdefault/\swdefault} {'bold~ swash'~}
595 } (#1/#2)~
596 with~ NFSS~ spec.:~
597 \l_@@_nfss_tl
598 \exp_not:n { \ }
599 -- \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
600 {
601   {\mddefault/\scdefault} {'small~ caps'~}
602   {\bfdefault/\scdefault} {'bold~ small~ caps'~}
603   {\mddefault/\scitdefault} {'italic~ small~ caps'~}
604   {\bfdefault/\scitdefault} {'bold~ italic~ small~ caps'~}
605   {\mddefault/\scsldefault} {'slanted~ small~ caps'~}
606   {\bfdefault/\scsldefault} {'bold~ slanted~ small~ caps'~}
607 }~( #1 / \@@_combo_sc_shape:n {#2} )~
608 with~ NFSS~ spec.:~
609 \l_@@_nfss_sc_tl
610 \tl_if_empty:IF {\l_@@_postadjust_tl}
611 {
612   \exp_not:N \ and~ font~ adjustment~ code:
613   \exp_not:N \ \l_@@_postadjust_tl
614 }
615 }
616 }

```

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 617 \tl_set:Nn \l_@@_pre_feat_sclist
618 (*XE)
619 {
620   \bool_if:NT \l_@@_ot_bool
621   {
622     \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }
623     \tl_if_empty:NF \l_@@_lang_tl { language = \l_@@_lang_tl ; }
624   }
625 }
626 (/XE)
627 (*LU)
628 {
629   mode = \l_@@_mode_tl ;
630   \tl_if_empty:NF \l_@@_shaper_tl { shaper = \l_@@_shaper_tl ; }
631   \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }

```

```

632     \tl_if_empty:NF \l_@@_lang_tl   { language = \l_@@_lang_tl   ; }
633   }
634 </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF 635 <LU>\cs_new:Nn \@@_make_smallcaps:TF
636 <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
637   {
638     \exp_args:No \@@_check_ot_feat:NnTF \l_@@_fontface_cs_tl {smcp} {#1} {#2}
639   }
640 <*XE>
641 \cs_new:Nn \@@_make_smallcaps:TF
642   {
643     \bool_if:NTF \l_@@_ot_bool
644       { \@@_make_ot_smallcaps:TF {#1} {#2} }
645     {
646       \bool_if:NT \l_@@_atsui_bool
647         {
648           \exp_args:No \@@_make_AAT_feature_string:NnnTF
649             \l_@@_fontface_cs_tl {3} {3} {#1} {#2}
650         }
651       }
652   }
653 </XE>

```

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

```

\@@_update_featstr:n 654 \cs_new:Nn \@@_update_featstr:n
655   {
656     <debug> \typeout{::: @@_update_featstr:n {#1}}
657     \bool_if:NF \l_@@_firsttime_bool
658     {
659       \tl_gset:Nx \g_@@_single_feat_tl { #1 }
660     <debug> \typeout{:::~ Adding~ feature.}
661     \tl_gput_right:Nx \g_@@_rawfeatures_sclist {#1;}
662   }
663 }

```

```

\@@_remove_clashing_featstr:n 664 \cs_new:Nn \@@_remove_clashing_featstr:n
665   {
666     <debug> \typeout{::: @@_remove_clashing_featstr:n {#1}}
667     \clist_map_inline:nn {#1}
668     {
669     <debug> \typeout{:::~ Removing~ feature~ "##1;}
670     \tl_gremove_all:Nn \g_@@_rawfeatures_sclist {##1;}
671   }
672 }
673 \cs_generate_variant:Nn \@@_remove_clashing_featstr:n {x}

```

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.)

```
\@@_init:
674 \cs_set:Npn \@@_init:
675   {
676   (debug) \typeout{:: @@_init:}
677     \bool_set_false:N \l_@@_ot_bool
678     \bool_set_true:N \l_@@_firsttime_bool
679     \@@_font_is_name:
680     \tl_clear:N \l_@@_font_path_tl
681     \tl_clear:N \l_@@_optical_size_tl
682     \tl_clear:N \l_@@_ttc_index_tl
683     \tl_clear:N \l_@@_renderer_tl
684     \tl_gclear:N \g_@@_defined_shapes_tl
685     \tl_gclear:N \g_@@_curr_series_tl
686     \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspec_encoding_tl
687   (*LU)
688     \tl_set:Nn \l_@@_mode_tl {node}
689     \int_set:Nn \prehyphenchar { \- } % fixe
690     \int_zero:N \posthyphenchar      % fixe
691     \int_zero:N \preexhyphenchar     % fixe
692     \int_zero:N \postexhyphenchar   % fixe
693   (/LU)
694   }
```

Executed in \@@_get_features:Nn.

```
\@@_init_fontface:
695 \cs_new:Nn \@@_init_fontface:
696   {
697     \tl_gclear:N \g_@@_rawfeatures_sclist
698     \tl_clear:N \l_@@_scale_tl
699     \tl_set_eq:NN \l_@@_opacity_tl \c_@@_opacity_tl
700     \tl_set_eq:NN \l_@@_hexcol_tl \c_@@_hexcol_tl
701     \tl_set_eq:NN \l_@@_postadjust_tl \c_@@_postadjust_tl
702     \tl_clear:N \l_@@_wordspace_adjust_tl
703     \tl_clear:N \l_@@_punctspace_adjust_tl
704   }
```

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

```
\@@_ot_validate_tag:n
705 (*LU)
706 \cs_new_protected:Nn \@@_ot_validate_tag:n
707   {
708     \@@_ot_validate_tag:w #1 \q_nil
709   }
710 \cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
711 \cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
712   {
713     \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
```

```

714     { \@@_ot_validate_tag_aux:w #2 \c_empty_tl \c_empty_tl \q_nil }
715     { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
716   }
717 \cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
718   {
719     \int_compare:nT { \tl_count:n {#5} > 2 }
720     { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
721   }
722 </LU>

```

`\@@_iv_str_to_num:Nn` This macro takes a four character string and converts it to the numerical representation required for Xe_{La}TeX 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.

```

723 <*XE>
724 \cs_new:Nn \@@_iv_str_to_num:Nn
725   {
726   (debug)\typeout{\iv_str_to_num:~#1~/~#2}
727     \@@_strip_leading_sign:Nw #1#2 \q_nil
728   }
729 \cs_generate_variant:Nn \@@_iv_str_to_num:Nn {Nx}

```

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

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

```

730 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
731   {
732     \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
733     { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
734     { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
735   }

```

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

```

736 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
737   {
738     \int_compare:nT { \tl_count:n {#6} > 2 }
739     { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }
740
741     \int_set:Nn #1
742     {
743       `#2 * "1000000
744       + `#3 * "10000
745       + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
746       + \ifx \c_empty_tl #5 32 \else `#5 \fi
747     }
748   }
749 </XE>

```

File XI

fontspec-code-opentype.dtx

1 OpenType definitions code

```
\@@_define_opentype_feature_group:n 1 \cs_new:Nn \@@_define_opentype_feature_group:n
2 {
3   \keys_define:nn {fontspec-opentype} { #1 .multichoice: , .groups:n = {opentype} }
4 }

#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)

5 \cs_new:Nn \@@_feat_prop_add:nn
6 {
7   \tl_if_empty:nF {#1}
8   {
9     \prop_if_in:NnF \g_@@_OT_features_prop {#1}
10    {
11      \prop_gput:Nnn \g_@@_OT_features_prop {#1} {#2}
12    }
13  }
14 }
15 \cs_new:Nn \@@_define_opentype_feature:nmnnn
16 {
17   \@@_feat_prop_add:nn {#3} {#1\,=#, #2}
18   \tl_if_empty:nTF {#4}
19   {
20     \keys_define:nn {fontspec-opentype}
21     {
22       #1/#2 .code:n =
23       { \@@_remove_clashing_featstr:n {#5} } ,
24       #1/#2 .groups:n = {opentype}
25     }
26   }
27   {
28     \keys_define:nn {fontspec-opentype}
29     {
30       #1/#2 .code:n =
31       {
32 (debug)       \typeout{::::::::::fontspec-opentype-#1/#2~::~#3/#4/#5}
33               \@@_make_OT_feature:nmn {#3} {#4} {#5}
34             } ,
35       #1/#2 .groups:n = {opentype}
36     }
37   }
```

```

37     }
38 }

#1 : Feature key
\@@_define_opentype_onoffreset:nmnn #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)
39 \cs_new:Nn \@@_feat_off:n {#10ff}
40 \cs_new:Nn \@@_feat_reset:n {#1Reset}
41 \cs_new:Nn \@@_define_opentype_onoffreset:nmnnn
42 {
43   \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#1} {#2} {#3} {+#4} {#5}
44   \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#1} { \@@_feat_off:n {#2} } {#3} {-#4}
45   \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#1} { \@@_feat_reset:n {#2} } {} {} {+#4},
46 }

#1 : Feature key
\@@_define_opentype_onreset:nmnnn #2 : Feature option val
#3 : Check feature
#4 : Exact tag string to activate
#5 : Tags to remove (clist)
47 \cs_new:Nn \@@_define_opentype_onreset:nmnnn
48 {
49   \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#1} {#2} {#3} {#4} {#5}
50   \exp_args:Nnx \@@_define_opentype_feature:nmnnn {#1} { \@@_feat_reset:n {#2} } {} {} {#4}
51 }

```

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.

```

52 \cs_new:Nn \@@_make_OT_feature:nnn
53 {
54 (debug) \typeout{: @@@_make_OT_feature:nnn \exp_not:n { {#1}{#2}{#3} } }
55
56   \bool_set_true:N \l_@@_proceed_bool
57
58   \tl_if_empty:nF {#1}
59   {
60     \exp_args:No \@@_check_ot_feat:NnF \l_@@_fontface_cs_tl {#1}
61     {
62       \@@_warning:nx {icu-feature-not-exist-in-font} {#1}
63       \bool_set_false:N \l_@@_proceed_bool
64     }

```

```

65     }
66
67     \@@_remove_clashing_featstr:x { #2 , \@@_swap_plus_minus:n {#2} , #3 }
68
69     \bool_if:NT \l_@@_proceed_bool { \@@_update_featstr:n {#2} }
70   }
71   \cs_generate_variant:Nn \@@_make_OT_feature:nnn {xxx}
72   \cs_new:Nn \@@_swap_plus_minus:n { \@@_swap_plus_minus_aux:Nq #1 \q_nil }
73   \cs_new:Npn \@@_swap_plus_minus_aux:Nq #1#2 \q_nil
74   { \str_case:nn {#1} { {+} {-#2} {-} {+#2} } }

```

(End 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.

```

75   \prg_new_conditional:Nnn \@@_check_script:Nn {TF,T}
76   {
77   <debug>\typeout{:: _check_script:Nn~#1~/~#2}
78     \bool_if:NnTF \l_@@_never_check_bool
79     { \prg_return_true: }
80     {
81     \bool_if:nTF { \tl_if_empty_p:e {#2} }
82     { \prg_return_false: }
83     {
84     <*XE>
85     <debug>\typeout{:::~ checking~ script~ #2}
86       \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
87       \int_set:Nn \l_tmpb_int { \XeTeXOTcountscritps #1 }
88       \int_zero:N \l_tmpa_int
89       \bool_set_false:N \l__fontspec_check_bool
90       \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
91       {
92       \ifnum \XeTeXOTscripttag #1 \l_tmpa_int = \l_@@_strnum_int
93       \bool_set_true:N \l__fontspec_check_bool
94       \int_set:Nn \l_tmpa_int { \l_tmpb_int }
95       \else
96       \int_incr:N \l_tmpa_int
97       \fi
98       }
99       \bool_if:NnTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
100     </XE>
101     <*LU>
102     \@@_ot_validate_tag:x {#2}
103     \cs_if_eq:NNTF #1 \font
104     { \tl_set:Nx \l_@@_tmp_tl { \curr@fontshape/\f@size } }
105     { \tl_set:Nx \l_@@_tmp_tl { \cs_to_str:N #1 } }
106     <debug>\typeout{:::~ checking:~"\l_@@_tmp_tl",~ "#2"}
107     \lua_now:e { fontspec.check_ot_script("\l_@@_tmp_tl", "#2") }
108     \bool_if:NnTF \l__fontspec_check_bool
109     {
110     <debug>\typeout{:::~ TRUE}

```



```

117         \prg_return_true:
118     }
119     {
120 (debug)\typeout{::::::~~ FALSE}
121         \prg_return_false:
122     }
123 </LU>
124 }
125 }
126 }

```

(End 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'.

```

127 \prg_new_conditional:Nnn \@@_check_lang:Nn {TF}
128 {
129     \@@_check_lang:NnnTF #1 {#2} {\l_@@_script_tl} {\prg_return_true:} {\prg_return_false:}
130 }
131
132 \prg_new_conditional:Nnn \@@_check_lang:Nnn {TF}
133 {
134 (debug)\typeout{: : _check_lang:Nn~#1~/~#2~/~#3~/}
135     \bool_if:NTF \l_@@_never_check_bool
136     { \prg_return_true: }
137     {
138         \bool_if:nTF { \tl_if_empty_p:e {#3} }
139         { \prg_return_false: }
140         {
141 (*XE)
142             \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
143             \@@_iv_str_to_num:Nx \l_@@_script_int {#3}
144             \int_set:Nn \l_@@_tmpb_int
145             { \XeTeXOTcountlanguages #1 \l_@@_script_int }
146             \int_zero:N \l_@@_tmpa_int
147             \bool_set_false:N \l_@@_fontspec_check_bool
148             \bool_until_do:nn { \int_compare_p:nNn \l_@@_tmpa_int = \l_@@_tmpb_int }
149             {
150                 \int_set:Nn \l_@@_tmpc_int
151                 { \XeTeXOTlanguagetag #1 \l_@@_script_int \l_@@_tmpa_int }
152
153                 \int_compare:nNnTF \l_@@_tmpc_int = \l_@@_strnum_int
154                 {
155                     \bool_set_true:N \l_@@_fontspec_check_bool
156                     \int_set:Nn \l_@@_tmpa_int {\l_@@_tmpb_int}
157                 }
158                 {
159                     \int_incr:N \l_@@_tmpa_int
160                 }
161             }
162         }
163     }
164 }

```

```

155     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
156 </XE>
157 (*LU)
158     \@@_ot_validate_tag:x {#2}
159     \@@_ot_validate_tag:x {#3}
160     \cs_if_eq:NNTF #1 \font
161       { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
162       { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
163     \@@_lua_function:neee {check_ot_lang} {\l_@@_tmp_tl} {#2} {#3}
164     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
165 </LU>
166   }
167 }
168 }

```

(End 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'.

```

169 \prg_new_conditional:Nnn \@@_check_ot_feat:Nn {TF,F}
170 {
171   \@@_check_ot_feat:NnnnTF #1 {#2} {\l_@@_lang_tl} {\l_@@_script_tl}
172   {\prg_return_true:} {\prg_return_false:}
173 }
174 \prg_new_conditional:Nnn \@@_check_ot_feat:Nnnn {TF,F}
175 {
176   \bool_if:NTF \l_@@_never_check_bool
177   { \prg_return_true: }
178   {
179     \bool_if:NTF { \tl_if_empty_p:e {#3} || \tl_if_empty_p:e {#4} }
180     { \prg_return_false: }
181     {
182 (*XE)
183 <debug>\typeout{::~ fontspec_check_ot_feat:nnn~ {#2}{#3}{#4}}
184     \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
185
186     \str_if_eq:eeTF {#3} {dflt}
187     { \int_zero:N \l_@@_language_int }
188     { \@@_iv_str_to_num:Nx \l_@@_language_int {#3} }
189     \@@_iv_str_to_num:Nx \l_@@_script_int {#4}
190
191     \int_set:Nn \l_tmpb_int
192     { \XeTeXOTcountfeatures #1 \l_@@_script_int \l_@@_language_int }
193
194     \int_zero:N \l_tmpa_int
195     \bool_set_false:N \l_@@_check_bool
196     \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
197     {
198       \ifnum\XeTeXOTfeaturetag #1 \l_@@_script_int \l_@@_language_int

```

```

200      \l_tmpa_int =\l_@@_strnum_int
201      \bool_set_true:N \l_@@_check_bool
202      \int_set:Nn \l_tmpa_int {\l_tmpb_int}
203    \else
204      \int_incr:N \l_tmpa_int
205    \fi
206  }
207  \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
208  </XE>
209  <*LU>
210  (debug)\typeout{::~~ fontspec_check_ot_feat:n~ {#1}}
211    \@@_ot_validate_tag:x {#2}
212    \@@_ot_validate_tag:x {#3}
213    \@@_ot_validate_tag:x {#4}
214    \cs_if_eq:NNTF #1 \font
215      { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
216      { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
217    \@@_lua_function:neeee {check_ot_feat} {\l_@@_tmp_tl} {#2} {#3} {#4}
218    \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
219  </LU>
220  }
221  }

```

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

1.2 OpenType feature information

```

222 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {aalt}{Access~All~Alternates}
223 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvf}{Above~base~Forms}
224 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvm}{Above~base~Mark~Positioning}
225 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvs}{Above~base~Substitutions}
226 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {afrc}{Alternative~Fractions}
227 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {akhn}{Akhands}
228 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwf}{Below~base~Forms}
229 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwm}{Below~base~Mark~Positioning}
230 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blws}{Below~base~Substitutions}
231 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {calt}{Contextual~Alternates}
232 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {case}{Case~Sensitive~Forms}
233 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ccmp}{Glyph~Composition~/~/Decomposition}
234 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cfar}{Conjunct~Form~After~Ro}
235 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cjit}{Conjunct~Forms}
236 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {clig}{Contextual~Ligatures}
237 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cpct}{Centered~CJK~Punctuation}
238 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {csp}{Capital~Spacing}
239 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {csw}{Contextual~Swash}
240 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {curs}{Cursive~Positioning}
241 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cvNN}{Character~Variant~$N$}
242 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2pc}{Petite~Capitals~From~Capitals}
243 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2sc}{Small~Capitals~From~Capitals}
244 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dist}{Distances}

```

245 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dlig}{Discretionary-Ligatures}
246 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dnom}{Denominators}
247 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dtls}{Dotless-Forms}
248 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {expt}{Expert-Forms}
249 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {falt}{Final-Glyph-on-Line-Alternates}
250 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin2}{Terminal-Forms-\#2}
251 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin3}{Terminal-Forms-\#3}
252 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fina}{Terminal-Forms}
253 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {flac}{Flattened-accent-forms}
254 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {frac}{Fractions}
255 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fwid}{Full-Widths}
256 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {half}{Half-Forms}
257 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {haln}{Halant-Forms}
258 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {halt}{Alternate-Half-Widths}
259 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hist}{Historical-Forms}
260 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hkna}{Horizontal-Kana-Alternates}
261 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hlig}{Historical-Ligatures}
262 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hngl}{Hangul}
263 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hojo}{Hojo-Kanji-Forms}
264 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hwid}{Half-Widths}
265 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {init}{Initial-Forms}
266 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {isol}{Isolated-Forms}
267 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ital}{Italics}
268 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jalt}{Justification-Alternates}
269 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp78}{JIS78-Forms}
270 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp83}{JIS83-Forms}
271 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp90}{JIS90-Forms}
272 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp04}{JIS2004-Forms}
273 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {kern}{Kerning}
274 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lfbd}{Left-Bounds}
275 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {liga}{Standard-Ligatures}
276 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ljmo}{Leading-Jamo-Forms}
277 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lnum}{Lining-Figures}
278 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {locl}{Localized-Forms}
279 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltra}{Left-to-right-alternates}
280 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltrm}{Left-to-right-mirrored-forms}
281 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mark}{Mark-Positioning}
282 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {med2}{Medial-Forms-\#2}
283 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {medi}{Medial-Forms}
284 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mgrk}{Mathematical-Greek}
285 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mkmk}{Mark-to-Mark-Positioning}
286 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mset}{Mark-Positioning-via-Substitution}
287 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nalt}{Alternate-Annotation-Forms}
288 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nlck}{NLC-Kanji-Forms}
289 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nukt}{Nukta-Forms}
290 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {numr}{Numerators}
291 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {onum}{Oldstyle-Figures}
292 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {opbd}{Optical-Bounds}
293 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ordn}{Ordinals}
294 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ornm}{Ornaments}
295 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {palt}{Proportional-Alternate-Widths}

296 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pcap}{Petite-Capitals}
297 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pkna}{Proportional-Kana}
298 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pnum}{Proportional-Figures}
299 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pref}{Pre-Base-Forms}
300 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pres}{Pre-base-Substitutions}
301 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstf}{Post-base-Forms}
302 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstst}{Post-base-Substitutions}
303 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pwid}{Proportional-Widths}
304 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {qwid}{Quarter-Widths}
305 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rand}{Randomize}
306 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rclt}{Required-Contextual-Alternates}
307 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rkrf}{Rakar-Forms}
308 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rlig}{Required-Ligatures}
309 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rphf}{Reph-Forms}
310 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtbd}{Right-Bounds}
311 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtla}{Right-to-left-alternates}
312 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtlm}{Right-to-left-mirrored-forms}
313 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ruby}{Ruby-Notation-Forms}
314 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rvrn}{Required-Variation-Alternates}
315 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {salt}{Stylistic-Alternates}
316 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sinf}{Scientific-Inferiors}
317 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {size}{Optical-size}
318 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smcp}{Small-Capitals}
319 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smpl}{Simplified-Forms}
320 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssNN}{Stylistic-Set- $\$N\$}$
321 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssty}{Math-script-style-alternates}
322 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {stch}{Stretching-Glyph-Decomposition}
323 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {subs}{Subscript}
324 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sup}{Superscript}
325 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {swsh}{Swash}
326 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {titl}{Titling}
327 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing-Jamo-Forms}
328 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional-Name-Forms}
329 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular-Figures}
330 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {trad}{Traditional-Forms}
331 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third-Widths}
332 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {unic}{Unicase}
333 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate-Vertical-Metrics}
334 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu-Variants}
335 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical-Writing}
336 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate-Vertical-Half-Metrics}
337 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vjmo}{Vowel-Jamo-Forms}
338 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical-Kana-Alternates}
339 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkrn}{Vertical-Kerning}
340 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional-Alternate-Vertical-Me
341 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrt2}{Vertical-Alternates-and-Rotation}
342 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrtr}{Vertical-Alternates-for-Rotation}
343 \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 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 definition for `\@@_define_aat_feature:nmnn`. This function is documented on page ??.)

`\@@_make_AAT_feature:nn`

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

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

`\@@_make_AAT_feature_string:NmnTF`

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

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

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

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

```

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

```

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

File XIII

fontspec-code-keyval.dtx

1 Font loading (keyval) definitions

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

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

Wrapper function to save some characters in the source:

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

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

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

1.1 Pre-pre-parsing stages

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

Don't load font config file

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

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

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



```

29     \tl_set:Nn \l_@@_font_path_tl {#1}
30     \@@_font_is_file:
31 < *XE >
32     \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
33 < /XE >
34   }
35   \aliasfontfeature{Path}{ExternalLocation}
36   \@@_keys_define_code:nnn {fontspec} {Path} {}

```

(End 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.

```

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

```

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.

```

47 < *XE >
48 \keys_define:nn {fontspec-renderer}
49 {
50   Renderer .choices:nn =
51     {AAT,ICU,OpenType,Graphite,Full,Basic,Node,Base,HarfBuzz,Harfbuzz}
52     {
53       \int_compare:nTF {\l_keys_choice_int <= 4}
54         {
55           \tl_set:Nx \l_@@_renderer_tl
56             {
57               \int_case:nn \l_keys_choice_int { 1 {/AAT} 2 {/OT} 3 {/OT} 4 {/GR} }
58             }
59 < debug > \typeout{Renderer:~ \l_@@_renderer_tl}
60           \tl_gset:Nx \g_@@_single_feat_tl { \l_@@_renderer_tl }
61         }
62     }
63     \@@_warning:nx {only-luatex-feature} {Renderer=Full/Basic/Node/Base/HarfBuzz}
64   }
65 }
66 < /XE >
67 < *LU >
68 \keys_define:nn {fontspec-renderer}
69 {
70   Renderer .choices:nn =

```

```

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

```

1.2 Pre-parsed features

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

```

104 \@@_keys_define_code:nnn {fontspec-prepare} {Script}
105 {
106 (XE) \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
107     \tl_set:Nn \l_@@_script_name_tl {#1}
108 }

```

Exactly the same:

```

109 \@@_keys_define_code:nnn {fontspec-prepare} {Language}
110 {
111 (XE) \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
112     \tl_set:Nn \l_@@_lang_name_tl {#1}
113 }

```

TTC font index

```
114 \@@_keys_define_code:nmn {fontspec-preparse} {FontIndex}
115   {
116     \str_if_eq:eeF { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
117     { \@@_warning:n {font-index-needs-ttc} }
118 (XE) \tl_set:Nn \l_@@_ttc_index_tl {:#1}
119 (LU) \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
120   }
121 \@@_keys_define_code:nmn {fontspec} {FontIndex}
122   {
123 (XE) \tl_set:Nn \l_@@_ttc_index_tl {:#1}
124 (LU) \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
125   }
```

1.3 Font faces

Upright

```
126 \@@_keys_define_code:nmn {fontspec-preparse-external} {UprightFont}
127   {
128     \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
129   }
```

Italic and slanted

```
130 \@@_keys_define_code:nmn {fontspec-preparse-external} {ItalicFont}
131   {
132     \tl_if_empty:nTF {#1}
133     {
134       \bool_set_true:N \l_@@_noit_bool
135     }
136     {
137       \bool_set_false:N \l_@@_noit_bool
138       \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
139     }
140   }
141 \@@_keys_define_code:nmn {fontspec-preparse-external} {SlantedFont}
142   {
143     \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
144   }
145 \@@_keys_define_code:nmn {fontspec-preparse-external} {SwashFont}
146   {
147     \fontspec_complete_fontname:Nn \l_@@_fontname_sw_tl {#1}
148   }
```

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!

```
149 %\@@_keys_define_code:nmn {fontspec-preparse-external} {BoldSeries}
150 % {
151 % \tl_gset:Nx \g_@@_curr_series_tl { #1 }
152 % \seq_put_right:Nx \l_@@_bf_series_seq { #1 }
```

153 % }

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

```
154 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldFont}
155 {
156   \tl_if_empty:nTF {#1}
157   {
158     \bool_set_true:N \l_@@_nobf_bool
159   }
160   {
161     \bool_set_false:N \l_@@_nobf_bool
162     \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
163
164     \seq_if_empty:NT \l_@@_bf_series_seq
165     {
166       \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
167       \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
168     }
169
170     \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
171     {
172       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
173     }
174
175     \prop_put:NxV \l_@@_nfss_prop {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
176
177     <debug>\typeout{Setting-bold-font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
178
179   }
180 }
```

Bold italic/slanted

```
181 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldItalicFont}
182 {
183   \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
184 }
185 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldSlantedFont}
186 {
187   \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
188 }
189 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldSwashFont}
190 {
191   \fontspec_complete_fontname:Nn \l_@@_fontname_bfsw_tl {#1}
192 }
```

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

```
193 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
194 {
195   \tl_if_empty:nTF {#1}
```

```

196     {
197     \bool_set_true:N \l_@@_nosc_bool
198     }
199     {
200     \bool_set_false:N \l_@@_nosc_bool
201     \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
202     }
203 }

```

1.3.1 Prepared font features

```

204 \@@_keys_define_code:nmn {fontspec-preparse} {UprightFeatures}
205 {
206   \clist_put_right:Nn \l_@@_fontfeat_up_clist {#1}
207 }
208 \@@_keys_define_code:nmn {fontspec-preparse} {BoldFeatures}
209 {
210   \clist_put_right:Nn \l_@@_fontfeat_bf_clist {#1}
211   % \prop_put:NxV \l_@@_nfss_prop
212   %   {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
213 }
214 \@@_keys_define_code:nmn {fontspec-preparse} {ItalicFeatures}
215 {
216   \clist_put_right:Nn \l_@@_fontfeat_it_clist {#1}
217 }
218 \@@_keys_define_code:nmn {fontspec-preparse} {BoldItalicFeatures}
219 {
220   \clist_put_right:Nn \l_@@_fontfeat_bfit_clist {#1}
221 }
222 \@@_keys_define_code:nmn {fontspec-preparse} {SlantedFeatures}
223 {
224   \clist_put_right:Nn \l_@@_fontfeat_sl_clist {#1}
225 }
226 \@@_keys_define_code:nmn {fontspec-preparse} {BoldSlantedFeatures}
227 {
228   \clist_put_right:Nn \l_@@_fontfeat_bfsl_clist {#1}
229 }
230 \@@_keys_define_code:nmn {fontspec-preparse} {SwashFeatures}
231 {
232   \clist_put_right:Nn \l_@@_fontfeat_sw_clist {#1}
233 }
234 \@@_keys_define_code:nmn {fontspec-preparse} {BoldSwashFeatures}
235 {
236   \clist_put_right:Nn \l_@@_fontfeat_bfsw_clist {#1}
237 }

```

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

```

238 \@@_keys_define_code:nmn {fontspec} {SmallCapsFeatures}
239 {
240   \bool_if:NF \l_@@_firsttime_bool
241   {
242     \clist_put_right:Nn \l_@@_fontfeat_sc_clist {#1}

```

```

243     }
244 }

Features varying by size

245 \@@_keys_define_code:nmn {fontspec-prepare} {SizeFeatures}
246 {
247     \clist_set:Nn \l_@@_sizefeat_clist {#1}
248     \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
249 }
250 \@@_keys_define_code:nmn {fontspec-prepare-nested} {SizeFeatures}
251 {
252     \clist_set:Nn \l_@@_sizefeat_clist {#1}
253     \tl_if_empty:NT \l_@@_this_font_tl
254     { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
255 }
256 \@@_keys_define_code:nmn {fontspec-prepare-nested} {Font}
257 {
258     \tl_set:Nn \l_@@_this_font_tl {#1}
259 }
260 \@@_keys_define_code:nmn {fontspec} {SizeFeatures}
261 {
262     % dummy
263 }
264 \@@_keys_define_code:nmn {fontspec} {Font}
265 {
266     % dummy
267 }

268 \@@_keys_define_code:nmn {fontspec-sizing} {Size}
269 {
270     \tl_set:Nn \l_@@_size_tl {#1}
271 }
272 \@@_keys_define_code:nmn {fontspec-sizing} {Font}
273 {
274     \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
275 }

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

276 \@@_keys_define_code:nmn {fontspec-opentype} {UprightFont} {}
277 \@@_keys_define_code:nmn {fontspec-opentype} {ItalicFont} {}
278 \@@_keys_define_code:nmn {fontspec-opentype} {SlantedFont} {}
279 \@@_keys_define_code:nmn {fontspec-opentype} {BoldFont} {}
280 \@@_keys_define_code:nmn {fontspec-opentype} {BoldItalicFont} {}
281 \@@_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.

```

282 \@@_keys_define_code:nmn {fontspec-prepare} {NFSSEncoding}
283 {

```

```

284 \tl_gset:Nx \g_@@_nfss_enc_tl { #1 }
285 }

```

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.)

```

286 \@@_keys_define_code:nmn {fontspec-preparse} {NFSSFamily}
287 {
288 \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
289 }

```

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

```

290 \@@_keys_define_code:nmn {fontspec-preparse} {FontFace}
291 {
292 \tl_clear:N \l_@@_this_font_tl
293 \clist_set:No \l_@@_arg_clist { \use_iii:nmn #1 }
294 \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
295 \int_compare:nT { \clist_count:N \l_@@_arg_clist = 1 }
296 {
297 <debug>\typeout{FontFace~ parsing:~ one~ clist~ item}
298 \tl_if_in:NnF \l_@@_arg_clist {=}
299 {
300 <debug>\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
301 \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
302 \tl_clear:N \l_@@_this_feat_clist
303 }
304 }
305
306 \@@_add_nfssfont:nmnn
307 { \use_i:nmn #1 } { \use_ii:nmn #1 } { \l_@@_this_font_tl } { \l_@@_this_feat_clist }
308 }

```

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

```

309 \@@_keys_define_code:nmn {fontspec} {Scale}
310 {
311 \str_case:nnF {#1}
312 {
313 {MatchLowercase} { \@@_calc_scale:n {5} }
314 {MatchUppercase} { \@@_calc_scale:n {8} }
315 }
316 { \tl_set:Nx \l_@@_scale_tl {#1} }
317 }

```

ScaleAgain

```

318 \@@_keys_define_code:nmn {fontspec} {ScaleAgain}
319 {
320 \tl_if_empty:NT \l_@@_scale_tl { \tl_set:Nn \l_@@_scale_tl {1} }
321 \tl_set:Nx \l_@@_scale_tl { \fp_eval:n { #1 * \l_@@_scale_tl } }
322 \@@_info:n {set-scale}

```

```
323 }
```

`\@@_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_YTeX).

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.)

```
324 \cs_new:Nn \@@_calc_scale:n
325 {
326   \group_begin:
327
328   \fontencoding { \encodingdefault }
329   \fontfamily { \familydefault }
330   \selectfont
331
332   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
333   \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_@@_fontface_cs_tl
334
335   \tl_set:Nx \l_@@_scale_tl
336   {
337     \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
338                 \dim_to_fp:n {\l_@@_tmpb_dim} }
339   }
340
341   \@@_info:n {set-scale}
342   \exp_args:NNNx
343   \group_end:
344   \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
345 }
```

(End definition for `\@@_calc_scale:n`. 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'.

```
346 \cs_new:Nn \@@_set_font_dimen:NnN
347 {
348   \dim_set:Nn #1 { \fontdimen #2 #3 }
349   \dim_compare:nNnT #1 = {0pt}
350   {
351     \settoheight #1
352     {
353       \str_if_eq:nnTF {#3} {\font} \rmfamily #3
354       \int_case:nnF #2
355       {
```



```

356         {5} {x} % x-height
357         {8} {X} % cap-height
358     } {?} % "else" clause; never reached.
359     }
360 }
361 }

```

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

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

```

362 \@@_keys_define_code:nmn {fontspec} {WordSpace}
363 {
364     \bool_if:NF \l_@@_firsttime_bool
365     { \_fontspec_parse_wordspace:w #1,,,\q_stop }
366 }
367 \@@_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}`.

```

368 \cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
369 {
370     \tl_if_empty:nTF {#4}
371     {
372         \tl_set:Nn \l_@@_wordspace_adjust_tl
373         {
374             \fontdimen 2 \font = #1 \fontdimen 2 \font
375             \fontdimen 3 \font = #1 \fontdimen 3 \font
376             \fontdimen 4 \font = #1 \fontdimen 4 \font
377         }
378     }
379     {
380         \tl_set:Nn \l_@@_wordspace_adjust_tl
381         {
382             \fontdimen 2 \font = #1 \fontdimen 2 \font
383             \fontdimen 3 \font = #2 \fontdimen 3 \font
384             \fontdimen 4 \font = #3 \fontdimen 4 \font
385         }
386     }
387 }

```

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

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

```

388 \@@_keys_define_code:nmn {fontspec} {PunctuationSpace}
389 {
390     \str_case_e:nnF {#1}
391     {
392         {WordSpace}
393         {
394             \tl_set:Nn \l_@@_punctspace_adjust_tl
395             { \fontdimen 7 \font = 0 \fontdimen 2 \font }

```

```

396     }
397     {TwiceWordSpace}
398     {
399     \tl_set:Nn \l_@@_punctspace_adjust_tl
400     { \fontdimen 7 \font = 1 \fontdimen 2 \font }
401     }
402   }
403   {
404   \tl_set:Nn \l_@@_punctspace_adjust_tl
405   { \fontdimen 7 \font = #1 \fontdimen 7 \font }
406   }
407 }
408 \@@_aff_error:n {PunctuationSpace}

```

Secret hook into the font-adjustment code

```

409 \@@_keys_define_code:nmn {fontspec} {FontAdjustment}
410 {
411   \tl_put_right:Nx \l_@@_postadjust_tl {#1}
412 }

```

Letterspacing

```

413 \@@_keys_define_code:nmn {fontspec} {LetterSpace}
414 {
415   \@@_update_featstr:n {letterspace=#1}
416 }

```

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

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

```

417 \@@_keys_define_code:nmn {fontspec} {HyphenChar}
418 {
419   \str_if_eq:nnTF {#1} {None}
420   {
421     \tl_put_right:Nn \l_@@_postadjust_tl
422     { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
423   }
424   {
425     (LU) \@@_warning:nx {only-xetex-feature} {HyphenChar}
426
427     \tl_if_single:nTF {#1}
428     { \tl_set:Nn \l_@@_hyphenchar_tl {`#1} }
429     { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }
430
431     \exp_args:No \@@_primitive_font_glyph_if_exist:NnTF \l_@@_fontface_cs_tl {\l_@@_hyphen
432     {
433       \tl_put_right:Nn \l_@@_postadjust_tl
434       { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
435     }

```

```

436         { \@@_error:nxx {no-glyph}{\l_fontspec_fontname_tl}{#1} }
437     }
438 }
439 }
440 \@@_aff_error:n {HyphenChar}

```

Color Hooks into pkgxcolor, which names its colours \color@<name>.

```

441 \@@_keys_define_code:nnn {fontspec} {Color}
442 {
443     \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
444     {
445         \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
446     }
447     {
448         \int_compare:nTF { \tl_count:n {#1} == 6 }
449         { \tl_set:Nn \l_@@_hexcol_tl {#1} }
450         {
451             \int_compare:nTF { \tl_count:n {#1} == 8 }
452             { \fontspec_parse_colour:viii #1 }
453             {
454                 \bool_if:NF \l_@@_firsttime_bool
455                 { \@@_warning:nx {bad-colour} {#1} }
456             }
457         }
458     }
459 }
460 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
461 {
462     \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
463     \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
464     {
465         \bool_if:NF \l_@@_firsttime_bool
466         { \@@_warning:nx {opa-twice-col} {#7#8} }
467     }
468     \tl_set:Nn \l_@@_opacity_tl {#7#8}
469 }
470 \aliasfontfeature{Color}{Colour}
471 \@@_keys_define_code:nnn {fontspec} {Opacity}
472 {
473     \int_set:Nn \l_@@_tmp_int {255}
474     \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
475     \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
476     {
477         \bool_if:NF \l_@@_firsttime_bool
478         { \@@_warning:nx {opa-twice} {#1} }
479     }
480     \tl_set:Nx \l_@@_opacity_tl
481     {
482         \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
483         \int_to_hex:n { \l_@@_tmp_int }

```

```

484     }
485 }

```

Mapping

```

486 (*XE)
487 \@@_keys_define_code:nmn {fontspec-aat} {Mapping}
488 {
489   \tl_set:Nn \l_@@_mapping_tl { #1 }
490 }
491 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
492 {
493   \tl_set:Nn \l_@@_mapping_tl { #1 }
494 }
495 </XE>
496 (*LU)
497 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
498 {
499   \str_if_eq:nnTF {#1} {tex-text}
500   {
501     \@@_warning:n {no-mapping-ligtext}
502     \msg_redirect_name:nmn {fontspec} {no-mapping-ligtext} {none}
503     \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
504   }
505   { \@@_warning:n {no-mapping} }
506 }
507 </LU>

```

1.4.1 Continuous font axes

```

508 \@@_keys_define_code:nmn {fontspec} {Weight}
509 {
510   \@@_update_featstr:n{weight=#1}
511 }
512 \@@_keys_define_code:nmn {fontspec} {Width}
513 {
514   \@@_update_featstr:n{width=#1}
515 }
516 \@@_keys_define_code:nmn {fontspec} {OpticalSize}
517 (*XE)
518 {
519   \bool_if:NTF \l_@@_ot_bool
520   {
521     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
522   }
523   {
524     \bool_if:NT \l_@@_mm_bool
525     {
526       \@@_update_featstr:n { optical size = #1 }
527     }
528   }
529   \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }

```

```

530     {
531       \bool_if:NT \l_@@_firsttime_bool
532       { \@@_warning:nx {no-opticals} {\l_fontspeg_fontname_tl} }
533     }
534   }
535 </XE>
536 <*LU>
537   {
538     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
539   }
540 </LU>

```

1.4.2 Font transformations

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

```

541 \keys_define:nn {fontspec}
542   {
543     FakeSlant .code:n =
544       {
545         \@@_update_featstr:n {slant=#1}
546       },
547     FakeSlant .default:n = {0.2}
548   }
549 \keys_define:nn {fontspec}
550   {
551     FakeStretch .code:n =
552       {
553         \@@_update_featstr:n {extend=#1}
554       },
555     FakeStretch .default:n = {1.2}
556   }
557 \keys_define:nn {fontspec}
558   {
559     FakeBold .code:n =
560       {
561         \@@_update_featstr:n {embolden=#1}
562       },
563     FakeBold .default:n = {1.5}
564   }

```

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.)

```

565 \keys_define:nn {fontspec}
566   {
567     AutoFakeSlant .code:n =
568       {
569         \bool_if:NT \l_@@_firsttime_bool

```

```

570     {
571       \tl_set:Nn \l_@@_fake_slant_tl {#1}
572       \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
573       \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspeg_fontname_tl
574       \bool_set_false:N \l_@@_noit_bool
575
576       \tl_if_empty:NF \l_@@_fake_embolden_tl
577         {
578           \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
579             {FakeBold=\l_@@_fake_embolden_tl}
580           \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
581           \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspeg_fontname_tl
582         }
583     }
584 },
585 AutoFakeSlant .default:n = {0.2}
586 }

```

Same but reversed:

```

587 \keys_define:nn {fontspec}
588 {
589   AutoFakeBold .code:n =
590     {
591       \bool_if:NT \l_@@_firsttime_bool
592         {
593           \tl_set:Nn \l_@@_fake_embolden_tl {#1}
594           \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
595           \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspeg_fontname_tl
596           \bool_set_false:N \l_@@_nobf_bool
597
598           \tl_if_empty:NF \l_@@_fake_slant_tl
599             {
600               \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
601                 {FakeSlant=\l_@@_fake_slant_tl}
602               \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
603               \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspeg_fontname_tl
604             }
605         }
606     },
607   AutoFakeBold .default:n = {1.5}
608 }

```

1.4.3 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.

```

609 \@@_keys_define_code:nmn {fontspec-opentype} {RawFeature}
610 {
611   \@@_update_featstr:n {#1}
612 }
613 \@@_keys_define_code:nmn {fontspec-aat} {RawFeature}
614 {

```

```
615     \@@_update_featstr:n {#1}  
616   }
```

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

Emulate CM extra ligatures.

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

2.2 Letters

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



```

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

```

2.3 Numbers

```

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

```

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

```

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

```

2.4 Vertical position

```

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

```

2.5 Contextuals

```

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

```

2.6 Diacritics

```

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

```

2.7 Kerning

```

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

```

2.8 Fractions

```

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

```

```

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

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

```

2.9 Style

```

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

```

2.10 CJK shape

```

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

```

```

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

```

2.11 Character width

```

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

```

2.12 Vertical

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

```

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

```

3 OpenType features that need numbering

3.1 Alternate

```

203 \@@_define_opentype_feature_group:n {Alternate}

```

```

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

```

3.2 Variant / StylisticSet

```

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

```

3.3 CharacterVariant

```

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

```

```

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

```

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

3.4 Annotation

```

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

```

3.5 Ornament

```

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

```

4 Script and Language

4.1 Script

```

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

```

```

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

```

4.2 Language

```

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

```

```

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

```

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

```

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

```

5 Backwards compatibility

```

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

```


File XV

fontspec-code-scripts.dtx

1 Font script definitions

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

```

45 \newfontscript{Grantha}{gran}
46 \newfontscript{Greek}{grek}
47 \newfontscript{Gujarati}{gjr2,gujr}
48 \newfontscript{Gunjala-Gondi}{gong}
49 \newfontscript{Gurmukhi}{gur2,guru}
50 \newfontscript{Hangul-Jamo}{jamo}
51 \newfontscript{Hangul}{hang}
52 \newfontscript{Hanifi-Rohingya}{rohng}
53 \newfontscript{Hanunoo}{hano}
54 \newfontscript{Hatran}{hatr}
55 \newfontscript{Hebrew}{hebr}
56 \newfontscript{Hiragana-and-Katakana}{kana}
57 \newfontscript{Imperial-Aramaic}{armi}
58 \newfontscript{Inscriptional-Pahlavi}{phli}
59 \newfontscript{Inscriptional-Parthian}{prti}
60 \newfontscript{Javanese}{java}
61 \newfontscript{Kaithi}{kthi}
62 \newfontscript{Kannada}{knd2,knda}
63 \newfontscript{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{Manichaeen}{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{Nandinagari}{nand}
 101 \newfontscript{Newa}{newa}
 102 \newfontscript{Nushu}{nshu}
 103 \newfontscript{Nyiakeng~Puachue~Hmong}{hmnp}
 104 \newfontscript{Odia}{ory2,orya}
 105 \newfontscript{Ogham}{ogam}
 106 \newfontscript{Ol~Chiki}{olck}
 107 \newfontscript{Old-Italic}{ital}
 108 \newfontscript{Old-Hungarian}{hung}
 109 \newfontscript{Old-North-Arabian}{narb}
 110 \newfontscript{Old-Permic}{perm}
 111 \newfontscript{Old-Persian-Cuneiform}{xpeo}
 112 \newfontscript{Old-Sogdian}{sogo}
 113 \newfontscript{Old-South-Arabian}{sarb}
 114 \newfontscript{Old-Turkic}{orkh}
 115 \newfontscript{Old-Uyghur}{ougr}
 116 \newfontscript{Osage}{osge}
 117 \newfontscript{Osmanya}{osma}
 118 \newfontscript{Pahawh~Hmong}{hmng}
 119 \newfontscript{Palmyrene}{palm}
 120 \newfontscript{Pau-Cin-Hau}{pauc}
 121 \newfontscript{Phags-pa}{phag}
 122 \newfontscript{Phoenician}{phnx}
 123 \newfontscript{Psalter-Pahlavi}{phlp}
 124 \newfontscript{Rejang}{rjng}
 125 \newfontscript{Runic}{runr}
 126 \newfontscript{Samaritan}{samr}
 127 \newfontscript{Saurashtra}{saur}
 128 \newfontscript{Sharada}{shrd}
 129 \newfontscript{Shavian}{shaw}
 130 \newfontscript{Siddham}{sidd}
 131 \newfontscript{Sign-Writing}{sgnw}
 132 \newfontscript{Sinhala}{sinh}
 133 \newfontscript{Sogdian}{sogd}
 134 \newfontscript{Sora-Sompeng}{sora}
 135 \newfontscript{Sumero-Akkadian-Cuneiform}{xsux}
 136 \newfontscript{Sundanese}{sund}
 137 \newfontscript{Syloti-Nagri}{sylo}
 138 \newfontscript{Syriac}{syrc}
 139 \newfontscript{Tagalog}{tglg}
 140 \newfontscript{Tagbanwa}{tagb}
 141 \newfontscript{Tai~Le}{tale}
 142 \newfontscript{Tai~Lu}{talu}
 143 \newfontscript{Tai~Tham}{lana}
 144 \newfontscript{Tai~Viet}{tavt}
 145 \newfontscript{Takri}{takr}
 146 \newfontscript{Tamil}{tml2,taml}

```

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

```

For convenience or backwards compatibility:

```

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

```

File XVI

fontspec-code-lang.dtx

1 Font language definitions

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

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

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

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

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

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

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

```

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

```

File XVII

fontspec-code-feat-aat.dtx

1 AAT feature definitions

These are only defined for X₃TeX.

1.1 Ligatures

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

Emulate CM extra ligatures.

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

1.2 Letters

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

1.3 Numbers

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

Both naming conventions are offered to select the number case.

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

1.4 Contextuals

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

1.5 Diacritics

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

1.6 Vertical position

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

1.7 Fractions

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

1.8 Alternate

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

```

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

```

1.9 Variant / StylisticSet

```

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

```

1.10 Style

```

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

```

1.11 CJK shape

```

108 \@@_define_aat_feature_group:n {CJKShape}
109 \@@_define_aat_feature:nmnn {CJKShape} {Traditional} {20} {0}
110 \@@_define_aat_feature:nmnn {CJKShape} {Simplified} {20} {1}
111 \@@_define_aat_feature:nmnn {CJKShape} {JIS1978} {20} {2}
112 \@@_define_aat_feature:nmnn {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 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 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 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 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 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 definition for `\DeclareUnicodeEncoding`. This function is documented on page ??.)

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

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

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

File XIX

fontspec-code-math.dtx

1 Selecting maths fonts

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

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

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

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

```

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

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

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

```

```

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

```

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

```

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

```

($3A_{16} = 58_{10}$) 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 L^AT_EX only generally defines one level of boldness, we omit `\mathbf` in the bold maths series. It can be specified as per usual with `\setboldmathrm`, which stores the appropriate family name in `\g_@@_bfmathrm_tl`.

```

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

```

```

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

```

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

File XX

fontspec-code-closing.dtx

1 Closing code

1.1 Finishing up

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

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

File XXI

fontspec-code-xfss.dtx

1 Changes to the NFSS

1 (*fontspec)

1.1 Italic small caps and so on

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

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

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

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

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

6 \cs_new:Nn \@@_merge_default_shapes:

7 {

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

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

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

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

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

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

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

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

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

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

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

19 }

20 \@@_merge_default_shapes:

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

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

22 {

23 \bool_lazy_and:nnTF

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

25 {

26 \cs_if_exist_p:c

27 {

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

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

30 }

31 }

32 \prg_return_true: \prg_return_false:

33 }

34 \cs_set_eq:NN \emfontdeclare \DeclareEmphSequence

1.2 Strong emphasis

\strongfontdeclare

```

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

```

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

`\strongenv`

```

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

```

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

`\strong`
`\strongreset`

```

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

```

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

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

```

74 \cs_set:Npn \reset@font

```

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

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

Programmer's interface for setting nesting levels:

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

Defaults:

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

```
81 </fontspec>
```

Index

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

Symbols	
\#	250, 251, 282
\,	1, 2, 3, 4, 5, 17
\-	689
@@ commands:	
\@@_DeclareFontShape:nnnnn	503, 510, <u>520</u> , 538, 551, 559, 572
\g_@@_OT_features_prop	9, 11, 73
\@@_add_nfssfont:nnnn	304, 305, 306, 306, 307, 308, 309, 310, 311, <u>325</u>
\@@_aff_error:n	11, 367, 408, 440
\l_@@_alias_bool	22, 206, 213, 219, 224, 231, 251
\l_@@_all_features_clist	21, 53, 98, 108, 122, 181, 270
\g_@@_all_keyval_modules_clist	1, 47, 208, 226
\g_@@_all_opentype_feature_names_prop	74, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343
\l_@@_arg_clist	59, 293, 294, 295, 298, 301
\l_@@_atsui_bool	7, 10, 203, 344, 353, 646
\l_@@_basename_tl	42, 10, 90, 322
\l_@@_bf_series_seq	45, 152, 164, 167
\g_@@_bfmathrm_tl	71, 72, 82, 87, 88, 89, 121
\@@_calc_scale:n	313, 314, <u>324</u>
\g_@@_cfg_bool	1, 7, 8, 17
\l_@@_check_bool	5, 65, 66, 195, 200, 206, 217
\@@_check_lang:Nn	121
\@@_check_lang:NnTF	97, <u>121</u> , 337
\@@_check_lang:Nnn	125
\@@_check_lang:NnnTF	110, <u>121</u>
\@@_check_ot_feat:Nn	169
\@@_check_ot_feat:NnTF	50, 60, <u>169</u> , 638
\@@_check_ot_feat:Nnnn	174
\@@_check_ot_feat:NnnnTF	67, <u>169</u>
\@@_check_script:Nn	75
\@@_check_script:NnTF	75, 80, 291
\@@_combo_sc_shape:n	511, 514, 560, 599, 607
\@@_construct_font_call:mn	135, 137, 141, 144, <u>150</u> , 176, 273, 391, 392, 414, 497
\@@_construct_font_call:nnnnn	150, 159
\l_@@_curr_bfname_tl	93, 162, 172, 175, 177, 212
\l_@@_curr_fontname_tl	92, 316, 317
\g_@@_curr_series_tl	91, 151, 166, 170, 175, 177, 212, 685
\@@_declare_shape:nnnn	402, <u>422</u>
\@@_declare_shape_loginfo:nn	438, <u>580</u>
\@@_declare_shape_slanted:nn	436, <u>530</u>
\@@_declare_shapes_bx:nn	437, <u>542</u>
\@@_declare_shapes_normal:nn	434, <u>501</u>
\@@_declare_shapes_smcaps:nn	435, <u>506</u>
\g_@@_default_fontopts_clist	46, 110, 121
\@@_define_aat_feature:nnnn	2, 3, 4, <u>5</u> , 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 60, 61, 62, 63, 65, 66, 67, 101, 102, 103, 104, 105, 106, 107, 109, 110, 111, 112, 113, 114, 115, 117, 118, 119, 120, 121, 122, 123, 124, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 185

\@@_define_aat_feature_group:n .	\l_@@_family_fontopts_clist
. 1 , 1 , 29 , 35 , 44 , 55 , 52 , 104 , 105 , 111
59 , 64 , 68 , 80 , 91 , 100 , 108 , 116 , 125 , 180	\g_@@_family_int_prop 78 , 247 , 253
\@@_define_opentype_feature:nnnnn	\l_@@_family_label_tl
. 5 , 7 , 29 , 42 , 104 , 106 , 127 , 150 , 164
43 , 44 , 45 , 49 , 50 , 61 , 77 , 93 , 105 , 111 ,	\@@_feat_off:n 39 , 44
112 , 113 , 115 , 120 , 121 , 122 , 125 ,	\@@_feat_prop_add:nn . 1 , 2 , 3 , 4 , 5 , 5 , 17
126 , 127 , 129 , 153 , 154 , 157 , 177 , 195	\@@_feat_reset:n 40 , 45 , 50
\@@_define_opentype_feature_-	\@@_find_autofonts: 261 , 281
group:n 1 , 6 , 28 , 41 ,	\l_@@_firsttime_bool
60 , 76 , 92 , 104 , 114 , 124 , 128 , 156 , 1 , 29 , 178 , 240 , 273 , 364 ,
176 , 190 , 194 , 203 , 222 , 236 , 258 , 268	454 , 465 , 477 , 531 , 569 , 591 , 657 , 678
\@@_define_opentype_onoffreset:nnnnn	\@@_font_is_file: 30 , 167 , 175
13 , 14 , 15 , 16 , 17 , 18 , 27 , 34 , 35 , 36 ,	\@@_font_is_name: 167 , 679
37 , 38 , 39 , 39 , 40 , 51 , 52 , 53 , 54 , 55 ,	\l_@@_font_path_tl 29 , 96 , 173 , 680
59 , 70 , 71 , 72 , 73 , 74 , 75 , 86 , 87 , 88 ,	\@@_font_suppress_not_found_-
89 , 90 , 91 , 100 , 101 , 102 , 103 , 110 ,	error: 5 , 9 , 38 , 269
123 , 142 , 143 , 144 , 145 , 146 , 147 ,	\l_@@_fontcfg_bool 12 , 13 , 18 , 22 , 81
148 , 149 , 150 , 151 , 152 , 155 , 168 ,	\l_@@_fontface_cs_tl 17 ,
169 , 170 , 171 , 172 , 173 , 174 , 175 ,	60 , 143 , 147 , 149 , 154 , 159 , 160 , 163 ,
187 , 188 , 189 , 190 , 191 , 192 , 193 ,	167 , 291 , 333 , 337 , 413 , 431 , 638 , 649
195 , 196 , 197 , 198 , 199 , 200 , 201 , 202	\l_@@_fontfeat_bf_clist
\@@_define_opentype_onreset:nnnnn 47 62 , 210 , 305 , 594
\g_@@_defined_shapes_tl	\l_@@_fontfeat_bfit_clist
. 88 , 184 , 582 , 684 64 , 220 , 309 , 578 , 580 , 600 , 602
\l_@@_defining_encoding_bool . . 3 ,	\l_@@_fontfeat_bfsl_clist 66 , 228 , 310
9 , 15 , 21 , 24 , 27 , 50 , 52 , 57 , 63 , 69 , 75	\l_@@_fontfeat_bfsw_clist 68 , 236 , 311
\l_@@_disable_defaults_bool 21 , 96 , 155	\l_@@_fontfeat_clist 57 , 129 , 192 , 274
\l_@@_em_int 36	\l_@@_fontfeat_curr_clist
\g_@@_em_normalise_slant_bool 26 58 , 457 , 466 , 479
\g_@@_em_prop 75	\l_@@_fontfeat_it_clist
\l_@@_em_switch_tl 113 63 , 216 , 306 , 572
\l_@@_em_tmp_tl 118	\l_@@_fontfeat_sc_clist . 69 , 242 , 457
\l_@@_emdef_int 37	\l_@@_fontfeat_sl_clist . 65 , 224 , 307
\l_@@_emshape_query_tl 112	\l_@@_fontfeat_sw_clist . 67 , 232 , 308
\@@_error:n 1 , 10 , 448	\l_@@_fontfeat_up_clist
\@@_error:nn 2 , 3 , 4 , 10 , 14 , 16 , 61 , 206 , 248 , 304
22 , 28 , 58 , 64 , 70 , 76 , 139 , 415 , 720 , 739	\g_@@_fontid_family_prop 77 , 227 , 255
\@@_error:nnn 436	\l_@@_fontid_tl
\g_@@_euenc_bool 9 , 10 , 19 , 23 , 36 , 39 , 58 42 , 21 , 23 , 97 , 223 , 227 , 235
\l_@@_ext_filename_tl 84 , 85 , 88 , 89 , 94	\l_@@_fontname_bf_tl
\l_@@_extension_tl 75 , 131 , 172 , 286 , 292 , 305 , 595
. 39 , 45 , 53 , 72 , 95 , 116 , 161	\l_@@_fontname_bfit_tl 76 ,
\l_@@_extensions_clist 48 , 50 , 61 , 256	133 , 183 , 285 , 286 , 287 , 309 , 581 , 603
\l_@@_external_bool 23 , 28 , 40 , 375	\l_@@_fontname_bfsl_tl
\@@_extract_all_features: 93 135 , 187 , 300 , 310
\@@_extract_all_features:n 20 , 93	\l_@@_fontname_bfsw_tl . 137 , 191 , 311
\l_@@_fake_embolden_tl	\l_@@_fontname_it_tl
. 129 , 576 , 579 , 593 74 , 132 , 138 , 285 , 297 , 306 , 573
\l_@@_fake_slant_tl . 128 , 571 , 598 , 601	\l_@@_fontname_sc_tl 138 , 201 , 461 , 473
	\l_@@_fontname_sl_tl 134 , 143 , 300 , 307

\l_@@_fontname_sw_tl . . .	136, 147, 308	\@@_keys_set_known:nnN	
\l_@@_fontname_tl . . .	98, 154, 156, 160	. . .	<u>67</u> , 121, 126, 128, 192, 194, 331, 400
\l_@@_fontname_up_tl	42, 45,	\l_@@_lang_name_tl	
	9, 125, 128, 130, 135, 137, 139, 141, 144	112, 131, 143, 144, 182, 184
\@@_fontname_wrap:n		\l_@@_lang_tl	
.	47, 152, 153, 169, 173	48, 142, 171, 279, 339, 356, 623, 632
\l_@@_fontopts_clist		\l_@@_language_int	
.	51, 101, 102, 112, 402, 410, 411, 412	30, 45, 187, 188, 192, 198, 277, 340, 357
\g_@@_fontopts_prop		\l_@@_leftover_clist	54, 400, 402
.	70, 86, 101, 104, 134, 137, 141, 142, 410	\@@_load_external_fontoptions:N	
\@@_get_features:Nn	60, <u>188</u>	18, <u>79</u> , 409
\@@_get_features:n	28, 188, 275, 486	\@@_load_font:	26, <u>131</u>
\l_@@_graphite_bool	11, 203, 347, 365	\@@_load_fontname:Nn	401, 405, 452, 473
\l_@@_harfbuzz_bool	10, 74, 97	\@@_lua_function:nn	<u>78</u>
\c_@@_hexcol_tl	154, 216, 700	\@@_lua_function:nnn	<u>78</u>
\l_@@_hexcol_tl		\@@_lua_function:nnnn	<u>78</u> , 163
.	147, 215, 217, 445, 449, 462, 700	\@@_lua_function:nnnnn	<u>78</u> , 216
\l_@@_hyphenchar_tl		\@@_main_DeclareFontExtensions:n	
.	146, 428, 429, 431, 434	122, <u>254</u>
\@@_if_autofont:nn	388	\@@_main_IfFontFeatureActiveTF:nnn	
\@@_if_autofont:nnTF	381	126, 259
\@@_if_detect_external:n	58	\@@_main_addfontfeatures:n	82, 86, 146
\@@_if_detect_external:nTF	12, <u>58</u> , 175	\@@_main_aliasfontfeature:nn	106, 203
\@@_if_font_feature:n	265	\@@_main_aliasfontfeatureoption:nnn	
\@@_if_font_feature:nTF	263	110, 222
\@@_if_merge_shape:n	21	\@@_main_fontspec:nn	<u>1</u> , 3
\@@_if_merge_shape:nTF	191	\@@_main_liningnums:n	137, 294
\@@_info:n	7, 133, 322, 341	\@@_main_newAATfeature:nnnn	94, 177
\@@_info:nn	8, 383	\@@_main_newfontface:NnnN	
\@@_info:nnn	9, 264	59, 63, 67, 71, <u>114</u>
\@@_init:	6, 174, 270, <u>674</u>	\@@_main_newfontfamily:NnnN	
\@@_init_fontface:	191, <u>695</u>	43, 47, 51, 55, <u>101</u> , 116
\@@_init_ttc:n	17, <u>70</u>	\@@_main_newfontfeature:nn	90, 170
\@@_int_mult_truncate:Nn	74, 474	\@@_main_newopentypefeature:nnn	
\@@_iv_str_to_num:Nn	98, 102, 187
.	86, 135, 136, 184, 188, 189, <u>723</u>	\@@_main_oldstylenums:n	132, 287
\@@_iv_str_to_num:w	733, 734, 736	\@@_main_setboldmathrm:nn	27, 69
\@@_keys_define_code:nnn		\@@_main_setmainfont:nn	8, 24, 39
.	7, 13, 16, 20, 24, 36, 37,	\@@_main_setmathrm:nn	23, 63
	46, 104, 109, 114, 121, 126, 130, 141,	\@@_main_setmathsf:nn	31, 75
	145, 149, 154, 181, 185, 189, 193,	\@@_main_setmathtt:nn	35, 81
	204, 208, 214, 218, 222, 226, 230,	\@@_main_setmonofont:nn	18, 50
	234, 238, 245, 250, 256, 260, 264,	\@@_main_setsansfont:nn	13, 37
	268, 272, 276, 277, 278, 279, 280,	\@@_make_AAT_feature:nn	9, <u>12</u> , 76, 87
	281, 282, 286, 290, 309, 318, 354,	\@@_make_AAT_feature_string:Nnn	26
	362, 388, 409, 413, 417, 441, 471,	\@@_make_AAT_feature_string:NnnTF	
	487, 491, 497, 508, 512, 516, 609, 613	12, 17, <u>26</u> , 648
\l_@@_keys_leftover_clist		\@@_make_OT_feature:nnn	
.	55, 123, 126, 127, 128,	33, 52, 71, 211, 218, 231, 242, 265, 275
	193, 194, 198, 199, 202, 204, 208, 209	\@@_make_font_shapes:Nnnnn	317, <u>397</u>

<code>\@@_make_ot_smallcaps:TF</code>	635	<code>\l_@@_optical_size_tl</code>	
<code>\@@_make_smallcaps:TF</code>	463 , 635 , 641	149 , 164 , 521 , 538 , 681
<code>\l_@@_mapping_tl</code>		<code>\l_@@_options_tl</code>	99 , 153 , 156 , 160
.	22 , 23 , 24 , 26 , 150 , 212 , 213 , 489 , 493	<code>\l_@@_ot_bool</code>	8 , 28 , 39 ,
<code>\g_@@_math_bool</code>	65 , 78 , 91 , 108 , 121 , 136 , 196 , 271 ,
.	5 , 6 , 18 , 98 , 100 , 101 , 102 , 103 ,	<code>\@@_ot_compat:nn</code>	345 , 361 , 370 , 519 , 529 , 620 , 643 , 677
.	104 , 105 , 106 , 107 , 110 , 112 , 113 ,	360 , 364 , 365 ,
.	116 , 118 , 119 , 120 , 121 , 122 , 123 ,	<code>\@@_ot_validate_tag:n</code>	366 , 367 , 368 , 369 , 370 , 371 , 372 ,
.	124 , 125 , 126 , 127 , 128 , 129 , 130 , 131	373 , 374 , 375 , 376 , 377 , 378 , 379 , 380
<code>\g_@@_math_euler_bool</code>	9 , 14 , 34	<code>\@@_ot_validate_tag:w</code>	708 , 711
<code>\g_@@_math_lucida_bool</code>		<code>\@@_ot_validate_tag_aux:w</code> 714 , 715 , 717	
.	13 , 14 , 15 , 15 , 40	<code>\g_@@_pkg_euler_loaded_bool</code> 2 , 3 , 8 , 16	
<code>\g_@@_mathrm_tl</code>	65 ,	<code>\c_@@_postadjust_tl</code>	156 , 701
.	66 , 74 , 75 , 77 , 78 , 81 , 84 , 98 , 120 , 124	<code>\l_@@_postadjust_tl</code>	
<code>\g_@@_mathsf_tl</code>	153 , 411 , 421 , 433 , 504 ,
.	77 , 78 , 79 , 91 , 99 , 122 , 125	511 , 539 , 554 , 562 , 575 , 610 , 613 , 701
<code>\g_@@_mathtt_tl</code>		<code>\l_@@_pre_feat_sclist</code>	137 ,
.	80 , 83 , 84 , 92 , 100 , 123 , 126	144 , 158 , 274 , 391 , 392 , 414 , 498 , 617
<code>\@@_merge_default_shapes:</code>	6 , 20	<code>\@@_preparse_features:</code>	25 , 117
<code>\l_@@_mm_bool</code>	9 , 346 , 357 , 524 , 529	<code>\l_@@_prev_unicode_name_tl</code> 48 , 53 , 101	
<code>\l_@@_mode_tl</code> 76 , 86 , 86 , 90 , 99 , 629 , 688		<code>\l_@@_primitive_font</code>	39 , 40
<code>\@@_msg_new:nn</code>		<code>\@@_primitive_font_current_name:</code>	
.	13 , 18 , 23 , 44 , 84 , 90 , 94 , 104 , 108 ,	56 , 184 , 186
.	112 , 117 , 122 , 127 , 133 , 137 , 143 ,	<code>\@@_primitive_font_get_name:N</code> 56 , 393	
.	147 , 151 , 156 , 160 , 165 , 169 , 174 ,	<code>\@@_primitive_font_glyph_if-</code>	
.	178 , 186 , 190 , 194 , 198 , 202 , 207 , 212	<code>exist:Nn</code>	44
<code>\@@_msg_new:nnn</code> 15 , 28 , 37 , 48 , 58 , 66 , 74		<code>\@@_primitive_font_glyph_if-</code>	
<code>\l_@@_never_check_bool</code>		<code>exist:NnTF</code>	44 , 431
.	28 , 78 , 128 , 176 , 272	<code>\@@_primitive_font_gset:Nnn</code>	1 , 26 , 28 , 33
<code>\g_@@_nfss_enc_tl</code>	4 ,	1 , 26 , 28 , 33
.	31 , 33 , 44 , 46 , 57 , 59 , 102 , 109 , 262 ,	<code>\@@_primitive_font_gset:NnnTF</code>	34
.	284 , 503 , 510 , 538 , 551 , 559 , 572 , 686	<code>\@@_primitive_font_gset:NnnTFTF</code>	21
<code>\l_@@_nfss_fam_tl</code>	106 , 225 , 240 , 288	<code>\@@_primitive_font_gset:Onn</code>	33 , 143
<code>\g_@@_nfss_family_tl</code>		<code>\@@_primitive_font_gset:OnnTF</code>	34
.	41 , 103 , 163 , 180 , 229 , 240 ,	<code>\@@_primitive_font_gset:p:NnnTF</code>	21
.	241 , 254 , 255 , 262 , 268 , 269 , 270 ,	<code>\@@_primitive_font_if_exist:nTF</code>	
.	271 , 276 , 277 , 278 , 279 , 503 , 510 ,	35 , 176
.	538 , 539 , 551 , 553 , 559 , 561 , 572 , 574	<code>\@@_primitive_font_if_null:nTF</code>	
<code>\l_@@_nfss_prop</code>	71 , 175 , 211	13 , 24 , 29 , 40
<code>\l_@@_nfss_sc_tl</code>		<code>\@@_primitive_font_if_null_p:N</code>	13
.	104 , 426 , 432 , 478 , 508 , 511 , 557 , 609	<code>\@@_primitive_font_set:Nnn</code>	1 , 21 , 23 , 31 , 39 , 391 , 392
<code>\l_@@_nfss_tl</code> 105 , 425 , 431 , 453 , 504 , 597		1 , 21 , 23 , 31 , 39 , 391 , 392
<code>\l_@@_nfssfont_prop</code>	72 , 312 , 335	<code>\@@_primitive_font_set:NnnTFTF</code>	21
<code>\l_@@_nobf_bool</code>		<code>\@@_primitive_font_set:Onn</code>	31
.	2 , 26 , 158 , 161 , 283 , 290 , 596	<code>\@@_primitive_font_set:OnnTF</code>	32 , 413
<code>\l_@@_noit_bool</code>		<code>\@@_primitive_font_set_hyphenchar:Nn</code>	
.	3 , 27 , 134 , 137 , 283 , 295 , 574	52 , 422 , 434
<code>\l_@@_nosc_bool</code> 4 , 197 , 200 , 459 , 470 , 476			
<code>\c_@@_opacity_tl</code> 155 , 216 , 463 , 475 , 699			
<code>\l_@@_opacity_tl</code>			
.	148 , 215 , 217 , 463 , 468 , 475 , 480 , 699		

\@@_primitive_font_set_p:NnnTF ..	21	\@@_setmathtt_hook:nn	85 , 92
\l_@@_proceed_bool	27 , 56 , 63 , 69	\@@_setmonofont_hook:nn	60 , 89
\l_@@_punctspace_adjust_tl	151 , 156 , 394 , 399 , 404 , 703	\@@_setsansfont_hook:nn	47 , 88
\g_@@_rawfeatures_sclist	59 , 157 , 274 , 278 , 487 , 498 , 661 , 670 , 697	\@@_setup_nfss:Nnnn	453 , 478 , 482
\@@_remove_clashing_featstr:n ..	23 , 67 , 664	\@@_setup_single_size:nn	429 , 441
\l_@@_renderer_tl	52 , 55 , 59 , 60 , 85 , 163 , 354 , 362 , 366 , 683	\l_@@_sffamily_encoding_tl	10 , 17 , 22 , 44 , 163
\l_@@_rmfamily_encoding_tl	9 , 15 , 21 , 31 , 162	\l_@@_sffamily_family_tl	42 , 43 , 160
\l_@@_rmfamily_family_tl	29 , 30 , 159	\@@_shape_merge:nn	5 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 24 , 29 , 193 , 516 , 517
\@@_sanitise_fontname:Nn	8 , 9 , 10 , 44 , 74 , 75 , 76 , 130 , 408	\l_@@_shaper_tl	81 , 86 , 87 , 91 , 100 , 630
\@@_save_family:nn	33 , 258	\g_@@_single_feat_tl	60 , 88 , 89 , 267 , 279 , 281 , 283 , 298 , 341 , 358 , 659
\@@_save_family_needed:n	219	\l_@@_size_tl	107 , 270 , 443 , 448 , 449 , 484 , 496
\@@_save_family_needed:nTF	31 , 219	\l_@@_sizedfont_tl	108 , 274 , 444 , 452 , 454
\@@_save_fontid_family:nn	235 , 245 , 257	\l_@@_sizefeat_clist	48 , 49 , 247 , 252 , 330 , 336
\@@_save_fontinfo:n	260 , 266	\l_@@_sizing_leftover_clist	56 , 447 , 453 , 479
\l_@@_saved_fontname_tl	100 , 427 , 444	\l_@@_smcp_shape_tl	III , 193 , 196 , 199 , 202
\l_@@_scale_tl	145 , 192 , 316 , 320 , 321 , 335 , 344 , 489 , 491 , 496 , 698	\@@_strip_leading_sign:Nw ..	727 , 730
\l_@@_script_int	29 , 42 , 94 , 136 , 138 , 144 , 189 , 192 , 198 , 276 , 296	\@@_strip_plus_minus:n	196 , 198
\l_@@_script_name_tl	107 , 131 , 140 , 141 , 141 , 180 , 182 , 183 , 294 , 308	\@@_strip_plus_minus_aux:Nq ..	198 , 199
\l_@@_script_tl	47 , 95 , 123 , 139 , 171 , 278 , 295 , 622 , 631	\l_@@_strnum_int	31 , 86 , 92 , 135 , 146 , 184 , 199 , 296 , 340
\l_@@_scriptlang_exist_bool	25 , 288 , 297 , 303 , 334 , 342 , 346	\l_@@_strong_int	38 , 55 , 58 , 59 , 62 , 64 , 67 , 77 , 79
\@@_select_font_family:nn	1 , 43 , 151 , 156 , 159 , 165	\g_@@_strong_prop	37 , 45 , 46 , 56 , 64 , 76
\@@_set_autofont:Nnn	285 , 286 , 287 , 292 , 297 , 300 , 373	\l_@@_strong_switch_tl	64 , 65 , 114
\@@_set_default_features:nn ..	76 , 118	\l_@@_strong_tmp_tl	56 , 58 , 119
\@@_set_faces:	263 , 302	\l_@@_strongdef_int ..	38 , 39 , 45 , 46 , 47
\@@_set_faces_aux:nnnn	312 , 314	\@@_swap_plus_minus:n	67 , 72
\@@_set_family:NnnN	147 , 156 , 157	\@@_swap_plus_minus_aux:Nq ..	72 , 73
\@@_set_font_default_features:nnn ..	77 , 123	\l_@@_test_font	136 , 142
\@@_set_font_dimen:NnN ..	332 , 333 , 346	\l_@@_tfm_bool	6 , 343 , 350
\@@_set_font_type:N	9 , 27 , 38 , 64 , 77 , 90 , 107 , 120 , 135 , 142 , 339	\l_@@_this_feat_clist ..	60 , 294 , 302 , 307
\@@_set_fontface:NnnN ..	161 , 169 , 170	\l_@@_this_font_tl	109 , 253 , 254 , 258 , 292 , 301 , 307 , 327 , 333 , 336
\@@_set_scriptlang:	27 , 175	\@@_tl_new_if_free:N	146 , 152
\@@_setboldmathrm_hook:nn ..	73 , 93	\l_@@_tmp_int ..	32 , 473 , 474 , 482 , 483
\@@_setmainfont_hook:nn	34 , 87	\l_@@_tmp_tl	41 , 42 , 44 , 45 , 93 , 94 , 104 , 105 , 106 , 107 , 115 , 123 , 124 , 129 , 130 , 134 , 137 , 138 , 139 , 141 , 142 , 152 , 161 , 162 , 163 , 214 , 215 , 216 , 227 , 229 , 233 , 234 , 235 , 247 , 249 , 250 , 252 , 253 , 254 , 331
\@@_setmathrm_hook:nn	67 , 90	\l_@@_tmpa_bool	20 , 63 , 66 , 68
\@@_setmathsf_hook:nn	79 , 91		

<code>\clist_new:N</code>	46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69
<code>\clist_put_left:Nn</code>	466
<code>\clist_put_right:Nn</code>	206, 210, 216, 220, 224, 228, 232, 236, 242, 248, 572, 578, 580, 594, 600, 602
<code>\clist_set:Nn</code>
<code>\clist_set_eq:NN</code>	294, 457
<code>\colon</code>	117, 30, 31
<code>\convertcolorspec</code>	445
cs commands:	
<code>\cs:w</code>	129
<code>\cs_end:</code>	129
<code>\cs_generate_variant:Nn</code>
<code>\cs_if_eq:NNTF</code>	103, 160, 213
<code>\cs_if_exist:NTF</code>	3, 7, 27, 199, 443
<code>\cs_if_exist_p:N</code>	26
<code>\cs_new:Nn</code>	1, 1, 1, 4, 5, 5, 5, 6, 7, 11, 12, 13, 15, 15, 24, 37, 39, 40, 41, 44, 47, 50, 52, 52, 63, 67, 69, 70, 72, 74, 75, 79, 79, 81, 93, 94, 101, 114, 117, 118, 123, 131, 146, 146, 147, 150, 156, 157, 157, 161, 167, 169, 170, 170, 171, 175, 177, 187, 188, 198, 203, 222, 245, 254, 258, 259, 266, 281, 283, 302, 314, 319, 324, 325, 330, 339, 346, 360, 373, 397, 405, 422, 441, 482, 501, 506, 514, 520, 530, 542, 580, 635, 636, 641, 654, 664, 695, 724
<code>\cs_new:Npn</code>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 57, 65, 66, 73, 199, 239
<code>\cs_new_eq:NN</code>	56
<code>\cs_new_protected:Nn</code>	1, 287, 294, 706
<code>\cs_set:Npn</code>	1, 5, 9, 31, 32, 33, 34, 61, 73, 74, 79, 80, 81, 82, 173, 368, 460, 674, 711, 717, 730, 736
<code>\cs_set_eq:NN</code>	34, 43, 60, 63, 87, 88, 89, 90, 91, 92, 93, 169, 180
<code>\cs_set_protected:Npn</code>	35
<code>\cs_to_str:N</code>	103, 105, 108, 129, 162, 215
<code>\cs_undefine:N</code>	77, 241, 525
<code>\cyrillicencoding</code>	51, 55, 83
D	
<code>\date</code>	55
<code>\ddot</code>	20
<code>\DeclareDocumentCommand</code>	1, 7, 13, 19, 25, 31, 51, 55, 61, 67, 67, 73
<code>\DeclareEmphSequence</code>	34
<code>\DeclareFontEncoding</code>	30, 33
<code>\DeclareFontExtensions</code>	120
<code>\DeclareFontFamily</code>	35, 262
<code>\DeclareFontSeriesDefault</code>
<code>\DeclareFontShape</code>
<code>\DeclareFontSubstitution</code>	31, 34
<code>\DeclareMathAccent</code>
<code>\DeclareMathDelimiter</code>	65, 66, 67, 68, 69
<code>\DeclareMathSymbol</code>
<code>\DeclareOption</code>	1, 5, 6, 7, 8, 9, 10, 11, 16
<code>\DeclareRobustCommand</code>	51
<code>\DeclareSymbolFont</code>	16, 74
<code>\DeclareSymbolFontAlphabet</code>	76
<code>\DeclareTextCommand</code>	5, 11
<code>\DeclareTextComposite</code>	23
<code>\DeclareTextCompositeCommand</code>	29
<code>\DeclareTextFontCommand</code>	72
<code>\DeclareTextSymbol</code>	17
<code>\DeclareUnicodeEncoding</code>	21, 31
<code>\def</code>	27
<code>\defaultfontfeatures</code>	73
<code>\Delta</code>	53
dim commands:	
<code>\dim_compare:nNnTF</code>	349
<code>\dim_eval:n</code>	3, 7
<code>\dim_new:N</code>	42, 43, 44
<code>\dim_set:Nn</code>	348
<code>\dim_to_fp:n</code>	337, 338
dim internal commands:	
<code>__dim_eval:w</code>	76
<code>__dim_eval_end:</code>	76
<code>\dot</code>	26
<code>\DTX</code>	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23
E	
<code>\else</code>	17, 95, 202, 745, 746
else commands:	
<code>\else:</code>	48
<code>\emfontdeclare</code>	34
<code>\EncodingAccent</code>	7
<code>\EncodingCommand</code>	1

<code>\EncodingComposite</code>	19
<code>\EncodingCompositeCommand</code>	25
<code>\encodingdefault</code> 80 , 21 , 22 , 23 , 33 , 46 , 59 , 328	
<code>\EncodingSymbol</code>	13
<code>\endinput</code>	8 , 13 , 22 , 28
<code>\ExecuteOptions</code>	21
exp commands:	
<code>\exp_after:wN</code>	446
<code>\exp_args:NNNx</code>	342
<code>\exp_args:Nnnx</code>	195
<code>\exp_args:Nnx</code>	43 , 44 , 45 , 49 , 50
<code>\exp_args:No</code>	17 , 60 , 88 , 291 , 337 , 431 , 638 , 648
<code>\exp_args:NV</code>	283
<code>\exp_args:Nx</code>	64 , 429
<code>\exp_args:Nxx</code>	185
<code>\exp_last_unbraced:No</code> ..	31 , 32 , 33 , 34
<code>\exp_not:N</code>	20 , 83 , 106 , 108 , 109 , 110 , 239 , 240 , 243 , 244 , 252 , 253 , 278 , 279 , 407 , 585 , 599 , 612 , 613
<code>\exp_not:n</code> ..	54 , 60 , 190 , 262 , 584 , 598
<code>\expandafter</code>	28
F	
<code>\familydefault</code>	32 , 45 , 58 , 329
<code>\fi</code>	19 , 28 , 28 , 32 , 41 , 54 , 97 , 98 , 110 , 116 , 204 , 358 , 367 , 745 , 746
fi commands:	
<code>\fi</code>	50
file commands:	
<code>\file_if_exist:nTF</code>	25 , 88
<code>\file_input:n</code>	89
<code>\filedate</code>	55
<code>\fileversion</code>	55
<code>\fmtname</code>	28
<code>\font</code> . 42 , 3 , 7 , 9 , 12 , 27 , 38 , 50 , 64 , 67 , 77 , 80 , 90 , 97 , 103 , 107 , 110 , 120 , 135 , 160 , 213 , 332 , 353 , 374 , 375 , 376 , 382 , 383 , 384 , 395 , 400 , 405 , 422 , 434	
<code>\fontdimen</code>	81 , 348 , 374 , 375 , 376 , 382 , 383 , 384 , 395 , 400 , 405
<code>\fontdimen8</code>	80
<code>\fontencoding</code> 4 , 9 , 10 , 11 , 15 , 17 , 19 , 109 , 328	
<code>\fontfamily</code>	30 , 14 , 15 , 16 , 17 , 18 , 19 , 108 , 163 , 329
<code>\fontname</code>	40 , 56
<code>\fontspec</code>	24 , 31 , 42 , 1
fontspec commands:	
<code>\fontspec_calc_scale:n</code>	79
<code>\fontspec_complete_fontname:Nn</code>	128 , 138 , 143 , 147 , 162 , 183 , 187 , 191 , 201 , 274 , 316 , 319
<code>\g_fontspec_default_fontopts_tl</code> .	31
<code>\g_fontspec_encoding_tl</code> 41 , 42 , 44 , 48 , 49 , 51 , 52 , 55 , 56 , 74 , 75 , 77 , 78 , 79 , 80 , 80 , 81 , 84 , 87 , 88 , 89 , 91 , 92 , 686	
<code>\l_fontspec_family_tl</code>	42 , 41 , 79 , 153 , 166
<code>\l_fontspec_feature_string_tl</code> 19 , 53	
<code>\l_fontspec_font</code> 42 , 149 , 154 , 160 , 167	
<code>\fontspec_font_if_exist:n</code>	171
<code>\fontspec_font_if_exist:nTF</code>	180
<code>\l_fontspec_fontname_tl</code> 42 , 45 , 8 , 18 , 21 , 81 , 101 , 115 , 120 , 125 , 125 , 130 , 140 , 192 , 273 , 287 , 292 , 297 , 304 , 309 , 401 , 419 , 424 , 427 , 436 , 452 , 473 , 484 , 497 , 532 , 573 , 581 , 595 , 603	
<code>\fontspec_gset_family:Nnn</code>	65 , 66 , 71 , 72 , 77 , 78 , 83 , 84 , 156
<code>\fontspec_gset_fontface:NNnn</code> ..	169
<code>\fontspec_if_aat_feature:nn</code>	5
<code>\fontspec_if_aat_feature:nnTF</code>	5
<code>\fontspec_if_current_feature:n</code> .	181
<code>\fontspec_if_current_feature:nTF</code>	181 , 283
<code>\fontspec_if_current_language:n</code> 131	
<code>\fontspec_if_current_language:nTF</code>	131
<code>\fontspec_if_current_script:n</code> ..	116
<code>\fontspec_if_current_script:nTF</code> 116	
<code>\fontspec_if_feature:n</code>	34
<code>\fontspec_if_feature:nnn</code>	60
<code>\fontspec_if_feature:nnnTF</code>	60
<code>\fontspec_if_feature:nTF</code>	34
<code>\fontspec_if_fontspec_font:</code>	1
<code>\fontspec_if_fontspec_font:TF</code> . 1 , 7 , 25 , 36 , 62 , 75 , 88 , 105 , 118 , 133 , 149	
<code>\fontspec_if_language:n</code>	86
<code>\fontspec_if_language:nn</code>	103
<code>\fontspec_if_language:nnTF</code>	103
<code>\fontspec_if_language:nTF</code>	86
<code>\fontspec_if_opentype:</code>	23
<code>\fontspec_if_opentype:TF</code>	23
<code>\fontspec_if_script:n</code>	73
<code>\fontspec_if_script:nTF</code>	73
<code>\fontspec_if_small_caps:</code>	189
<code>\fontspec_if_small_caps:TF</code>	189
<code>\fontspec_maybe_setup_maths:</code>	94
<code>\fontspec_new_lang:nn</code>	118 , 330

`\fontspec_new_script:nn` 114, 283
`\fontspec_parse_colour:niii` . 452, 460
`\fontspec_parse_cv:w` 239, 252
`_fontspec_parse_whitespace:w` 365, 368
`\fontspec_select:nn` 43
`\fontspec_set_family:Nnn`
. 3, 29, 42, 55, 103, 146
`\fontspec_set_fontface:NNnn` 159
`\fontspec_set_strong_level:n` 79
`\fontspec_setup_maths:` 1, 134
`\fontspec_tmp:` 60, 63
fontspec internal commands:
`\l__fontspec_check_bool`
. 89, 93, 99, 108, 140, 148, 155, 164
`_fontspec_update_featstr:n` 97
`\FONTSPECCTX` 2
`\FontspecSetCheckBoolFalse` 65
`\FontspecSetCheckBoolTrue` 65
fp commands:
`\fp_eval:n` 321, 337
`\fp_new:N` 40, 41

G

`\g` 118
`\Gamma` 52
`\gdef` 2
`\GetFileInfo` 54
`\global` 7
`\grave` 19
group commands:
`\group_begin:` 4, 28, 37, 40, 151,
173, 268, 289, 296, 326, 390, 399, 523
`\group_end:`
. 33, 39, 41, 42, 49, 162, 177, 178,
276, 292, 299, 343, 394, 395, 403, 526

H

`\hat` 116, 25
`\hbar` 71

I

`\IfBooleanTF` 120, 132
`\ifcase` 348
`\ifdefined` 26, 39, 52
`\IfFontExistsTF` 180
`\IfFontFeatureActiveTF` 124, 259
`\IfNoValueTF` 75
`\ifnum` 92, 198, 355
`\ifx` 15, 28, 30, 98, 110, 116, 745, 746
`\ignorespaces` 4, 9, 14, 19, 78, 168
`\InputIfFileExists` 3

int commands:
`\int_case:nn` 57, 78, 83
`\int_case:nnTF` 354
`\int_compare:nNnTF` 146
`\int_compare:nTF` 32,
53, 74, 100, 295, 448, 451, 482, 719, 738
`\int_compare_p:nNn` 90, 141, 196
`\int_eval:n` 250
`\int_if_even:nTF` 37
`\int_incr:N` 47, 62, 96, 152, 203
`\int_new:N`
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39
`\int_set:Nn` 11, 42,
45, 58, 76, 79, 84, 87, 94, 94, 137, 143,
149, 191, 201, 296, 340, 473, 689, 741
`\int_to_hex:n` 483
`\int_use:N` 46, 55, 59, 64
`\int_zero:N` 38, 67,
77, 88, 139, 187, 194, 357, 690, 691, 692
`\l_tmpa_int` 88,
90, 92, 94, 96, 194, 196, 199, 201, 203
`\l_tmpb_int` 87, 90, 94, 191, 196, 201
`\itdefault` 2, 9, 12, 15, 77, 84, 89, 306, 309,
534, 535, 539, 568, 569, 574, 589, 592

K

keys commands:
`\l_keys_choice_int` 53, 57, 74, 78, 83
`\keys_define:nn`
. 3, 3, 7, 9, 20, 20, 22, 28, 48, 69, 69,
81, 92, 172, 204, 214, 215, 223, 232,
239, 246, 246, 259, 269, 278, 285,
325, 332, 417, 541, 549, 557, 565, 587
`\keys_if_choice_exist:nnnTF` . 182, 192
`\keys_if_exist:nnTF`
. 179, 189, 210, 228, 236, 243
`\l_keys_key_tl` 114, 119, 124, 129
`\keys_set:nn` 8, 13, 32,
42, 85, 106, 111, 183, 184, 199, 204,
209, 215, 233, 240, 247, 319, 349, 503
`\keys_set_groups:nnn` 412
`\keys_set_known:nn` 15
`\keys_set_known:nnN` 70, 86, 152, 446
`\l_keys_value_tl` 114, 119, 124, 129

L

`\l` 31, 51, 54, 64–66, 71
`\Lambda` 55
`\latinencoding` 52, 56, 84
`\liningnums` 35, 135, 287
lua commands:
`\lua_now:n` 6, 79, 80, 81, 82, 107

<code>\LuaLaTeX</code>	33	<code>\newfontlanguage</code>	I,
			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, <u>116</u> , 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, 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
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>	118, 56, 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>	56, 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, 118, 76, 87		
<code>\mathsf</code>	79, 91		
<code>\mathtt</code>	80, 92		
<code>\mddefault</code>	74,		
	75, 77, 79, 80, 87, 89, 304, 306, 307,		
	308, 587, 589, 590, 591, 601, 603, 605		
<code>\mkern</code>	71		
msg commands:			
<code>\msg_error:nn</code>	1		
<code>\msg_error:nnn</code>	2, 3		
<code>\msg_fatal:nn</code>	21		
<code>\msg_info:nn</code>	7		
<code>\msg_info:nnn</code>	8		
<code>\msg_info:nnnn</code>	9		
<code>\msg_line_context:</code>	96		
<code>\msg_new:nnn</code>	11, 14, 15		
<code>\msg_new:nnnn</code>	12, 16		
<code>\msg_redirect_module:nnn</code>	13, 14, 18, 19		
<code>\msg_redirect_name:nnn</code>	502		
<code>\msg_trace:nn</code>	10		
<code>\msg_warning:nn</code>	4		
<code>\msg_warning:nnn</code>	5		
<code>\msg_warning:nnnn</code>	6		
N			
<code>\newAAfeature</code>	92, <u>177</u>		
<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>\newfontface</code>	30, 57		
<code>\newfontfamily</code>	41		
<code>\newfontfeature</code>	32, 88, <u>170</u>		

\newfontscript	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		
\newICUfeature	100		
\newopentypefeature	96, 187		
\normalfont	35, 41, 48, 61, 76		
\normalsize	50, 524		
\nullfont	15		
\numexpr	44		
O			
\oldstylenums	35, 128, 287		
\Omega	62		
\or	351, 359, 363		
P			
Path	24		
\Phi	60		
\Pi	57		
\postexhyphenchar	692		
\posthyphenchar	690		
\preexhyphenchar	691		
\prehyphenchar	689		
prg commands:			
\prg_new_conditional:Nnn	1, 5, 21, 23, 26, 34, 44, 58, 60, 73, 75, 86, 103, 116, 121, 125, 131, 169, 171, 174, 181, 189, 219, 265, 388		
\prg_return_false: 3, 13, 16, 18, 20, 24, 28, 29, 30, 31, 32, 41, 49, 50, 51, 53, 57, 67, 68, 69, 71, 80, 82, 82, 84, 97, 99, 99, 101, 110, 112, 114, 115, 123, 125, 127, 129, 132, 140, 142, 144, 155, 164, 172, 178, 180, 187, 204, 206, 207, 217, 230, 281, 284, 394			
\prg_return_true:	3, 13, 16, 24, 28, 29, 32, 42, 47, 50, 54, 67, 68, 79, 80, 97, 99, 110, 111, 123, 125, 129, 140, 155, 164, 172, 177, 177, 187, 205, 206, 217, 236, 242, 284, 395		
\prg_set_conditional:Nnn	13, 21, 26, 35		
\ProcessOptions	22		
prop commands:			
\prop_gc_clear:N	37		
\prop_get:NnN	41, 44, 47, 48, 93, 95, 123, 138, 153, 154		
\prop_get:NnNTF	56, 64, 91, 92, 101, 104, 134, 227, 247, 410		
\prop_gput:Nnn	11, 46, 90, 137, 142, 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, 253, 254, 255, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 269, 270, 270, 271, 271, 272, 273, 274, 275, 276, 276, 277, 277, 278, 278, 279, 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		
\prop_gput_if_new:Nnn	45, 89		
\prop_gremove:Nn	141		
\prop_if_in:NnTF	9, 86		
\prop_map_inline:Nn	312		
\prop_new:N 70, 71, 72, 73, 74, 75, 76, 77, 78, 268		
\prop_put:Nnn	87, 88, 175, 211, 335		
\providecommand	2, 3, 4		
\ProvideDocumentCommand	55, 71		
\providefontface	69		
\providefontfamily	53		
\ProvidesExplFile	48		
\ProvidesExplPackage	43, 44, 45		
\Psi	61		
Q			
\lquad	55		

quark commands:	
\q_nil	72, 73, 198, 199, 240, 253, 708, 711, 714, 715, 717, 727, 730, 733, 734, 736
\q_stop	365, 368
	R
\relax	29, 44, 355
\renewcommand	71
\RenewDocumentCommand	47, 63, 130
\renewfontface	61
\renewfontfamily	45
\RequirePackage	5, 42, 47, 48, 62
\RequirePackageWithOptions	7, 12
\rmdefault	29, 118, 30, 32, 45, 98, 124
\rmfamily	80, 7, 353
	S
scan commands:	
\scan_stop:	3, 7, 46, 54
\scdefault	2, 3, 4, 8, 9, 10, 11, 12, 13, 14, 17, 18, 516, 517, 518, 601, 602
\scitdefault	2, 9, 12, 15, 16, 17, 603, 604
\scsldefault	3, 10, 13, 15, 16, 18, 605, 606
\scswdefault	4, 11, 14
\select	19
\selectfont	5, 110, 163, 330
seq commands:	
\seq_if_empty:NTF	164
\seq_new:N	45
\seq_put_right:Nn	152, 167
\setboldmathrm	28, 118, 25, 69, 95
\setfontface	65
\setfontfamily	49
\setmainfont	24, 28, 116, 6, 24
\SetMathAlphabet	77, 78, 79, 80, 84, 87, 88, 89, 91, 92
\setmathrm	118, 21, 63, 94
\setmathsf	29, 75, 96
\setmathtt	33, 81, 97
\setmonofont	16, 50
\setromanfont	37
\setsansfont	11, 37
\SetSymbolFont	17, 75, 81
\settoheight	351
\sfdefault	43, 45, 46, 99, 125
\sffamily	7
\shapedefault	8, 17, 18, 74, 75, 78, 79, 80, 81, 87, 88, 91, 92, 203, 304, 305, 587, 588
\Sigma	58
\sldefault	3, 10, 13, 16, 307, 310, 535, 538, 569, 573, 590, 593
\space	34, 39, 44, 192
str commands:	
\c_backslash_str	78
\c_colon_str	240, 253
\str_case:nn	74, 585, 599
\str_case:nnTF	201, 311
\str_case_e:nnTF	390
\str_if_eq:nnTF	32, 45, 58, 72, 93, 116, 124, 139, 155, 186, 215, 353, 393, 419, 499
\str_if_eq_p:nn	306, 314, 315, 316, 534, 535, 546, 547, 568, 569, 713, 732
\str_lowercase:n	72, 116
\string	21, 56, 76, 96
\strong	72
\strongenv	51, 72
\strongfontdeclare	35, 80
\strongreset	42, 68, 72
\suppressfontnotfounderror	11
\swdefault	4, 11, 14, 308, 311, 591, 594
sys commands:	
\sys_if_engine luatex:TF	3
\sys_if_engine xetex:TF	10
	T
T _E X and L ^A T _E X 2 _ε commands:	
\@	31, 61
\@filelist	50
\@ifpackageloaded	1, 6, 13, 14, 15, 96, 100, 101, 102, 103, 104, 105, 106, 107, 108, 112, 113, 114, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130
\@nomath	53
\@onlypreamble	94, 95, 96, 97
\@rmfamilyhook	7, 9
\@sffamilyhook	10
\@tempa	29, 30
\@ttfamilyhook	11
\add@unicode@accent	11
\color@	443
\curr@fontshape	104, 161, 214
\define@antt@mathversions	98
\define@iwona@mathversions	110
\define@kurier@mathversions	116
\fontencoding	28, 199, 202, 203
\fontfamily	3, 3, 28, 41, 44, 47, 48, 93, 95, 123, 138, 153, 154, 199, 202, 203
\fontseries	28, 45, 56, 59, 199, 202, 203

<code>\f@shape</code>	24, 29, 193		
<code>\f@size</code>	39, 104,		
	138, 145, 161, 214, 391, 392, 414, 525		
<code>\reset@font</code>	74		
<code>\two@digits</code>	231, 243, 244		
tex commands:			
<code>\tex_font:D</code>	59		
<code>\tex_hyphenchar:D</code>	54		
<code>\tex_iffontchar:D</code>	46		
<code>\textsc</code>	36		
<code>\textsf</code>	32		
<code>\Theta</code>	54		
<code>\tilde</code>	21		
<code>\title</code>	31		
tl commands:			
<code>\c_empty_tl</code>			
	61, 714, 715, 733, 734, 745, 746		
<code>\tl_build_begin:N</code>	425, 426		
<code>\tl_build_end:N</code>	431, 432		
<code>\tl_build_put_right:Nn</code>	494		
<code>\tl_clear:N</code>			
	23, 24, 45, 106, 135, 292, 302, 308,		
	443, 680, 681, 682, 683, 698, 702, 703		
<code>\tl_clear_new:N</code>	82, 83, 84		
<code>\tl_const:Nn</code>	8, 9, 10, 11,		
	12, 13, 14, 15, 16, 17, 18, 154, 155, 156		
<code>\tl_count:n</code>	448, 451, 719, 738		
<code>\tl_gc_clear:N</code>	267, 684, 685, 697		
<code>\tl_gput_right:Nn</code>	582, 661		
<code>\tl_gremove_all:Nn</code>	670		
<code>\tl_gset:Nn</code>	41, 42,		
	44, 60, 88, 98, 99, 100, 124, 125, 126,		
	151, 166, 254, 284, 298, 341, 358, 659		
<code>\tl_gset_eq:NN</code> .	156, 169, 229, 240, 686		
<code>\tl_if_empty:NTF</code>			
	29, 46, 50, 82, 91, 180,		
	212, 225, 253, 281, 320, 333, 354,		
	362, 366, 379, 448, 461, 489, 508,		
	557, 576, 598, 622, 623, 630, 631, 632		
<code>\tl_if_empty:nTF</code> 7, 14, 18, 58, 95, 96,			
	97, 132, 140, 156, 195, 329, 370, 377, 610		
<code>\tl_if_empty_p:n</code>	81, 94, 131, 179		
<code>\tl_if_eq:NNTF</code>	201, 463, 475		
<code>\tl_if_eq:nnTF</code>	98, 170		
<code>\tl_if_exist:NTF</code>	146, 516		
<code>\tl_if_exist_p:N</code>	24		
<code>\tl_if_in:NnTF</code>	50, 50, 298		
<code>\tl_if_in:nnTF</code>	65, 185		
<code>\tl_if_single:nTF</code>	128, 427		
<code>\tl_new:N</code>	79,		
	80, 81, 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, 117, 118, 119, 120, 121, 122,		
	123, 127, 128, 129, 130, 131, 132,		
	133, 134, 135, 136, 137, 138, 139,		
	140, 142, 143, 145, 146, 146, 147,		
	148, 149, 150, 151, 152, 153, 157,		
	158, 159, 159, 160, 161, 162, 163, 164		
<code>\tl_put_left:Nn</code>	46		
<code>\tl_put_right:Nn</code>			
	9, 10, 11, 136, 411, 421, 433		
<code>\tl_remove_all:Nn</code> ...	47, 85, 234, 323		
<code>\tl_remove_once:Nn</code>	52		
<code>\tl_replace_all:Nnn</code> .	14, 16, 18, 99, 322		
<code>\tl_set:Nn</code>	21, 21, 22, 22, 23, 26,		
	28, 29, 34, 39, 39, 42, 45, 46, 46, 47,		
	49, 53, 53, 55, 76, 81, 84, 99, 100, 104,		
	105, 107, 112, 118, 119, 123, 124,		
	129, 141, 144, 149, 150, 154, 160,		
	161, 162, 163, 164, 167, 196, 214,		
	215, 233, 249, 252, 254, 258, 270,		
	288, 294, 295, 316, 320, 321, 321,		
	327, 335, 339, 344, 354, 356, 362,		
	366, 372, 380, 382, 394, 399, 404,		
	428, 429, 449, 462, 468, 480, 489,		
	491, 493, 521, 538, 571, 593, 617, 688		
<code>\tl_set_eq:NN</code> .	30, 31, 33, 41, 43, 44,		
	46, 48, 49, 51, 52, 53, 55, 56, 56, 57,		
	59, 125, 157, 170, 172, 193, 301, 427,		
	444, 573, 581, 595, 603, 699, 700, 701		
<code>\tl_to_str:N</code>	21		
<code>\tl_to_str:n</code>	78, 186		
<code>\tl_trim_spaces:n</code>	14, 16		
<code>\tl_use:N</code>	29, 517		
<code>\tmpa</code>	27, 28		
token commands:			
<code>\token_to_str:N</code>	78, 443		
<code>\ttdefault</code>	47, 56, 58, 100, 126		
<code>\ttfamily</code>	7		
<code>\typeout</code> ..	3, 5, 23, 32, 37, 54, 55, 59, 59,		
	60, 69, 71, 77, 83, 85, 86, 95, 106, 110,		
	114, 119, 125, 127, 133, 135, 141,		
	148, 156, 177, 177, 182, 183, 183,		
	184, 190, 198, 202, 205, 208, 209,		
	212, 222, 223, 230, 238, 245, 261,		
	262, 278, 279, 287, 293, 297, 300,		
	305, 341, 349, 352, 356, 360, 364,		
	407, 424, 449, 454, 465, 469, 484,		
	487, 522, 656, 660, 666, 669, 676, 726		

U	
<code>\UndeclareAccent</code>	55
<code>\UndeclareCommand</code>	55
<code>\UndeclareComposite</code>	73
<code>\UndeclareSymbol</code>	55
<code>\UndeclareTextCommand</code>	59, 65, 71
<code>\unexpanded</code>	95, 125
<code>\UnicodeEncodingName</code>	5,
	11, 17, 23, 29, 48, 49, 53, 59, 65, 71, 78
<code>\UnicodeFontFile</code>	38, 40, 42, 44, 46
<code>\UnicodeFontTeXLigatures</code>	38, 40, 42, 44, 46
<code>\Upsilon</code>	59
<code>\url</code>	39
use commands:	
<code>\use:N</code>	99, 108
<code>\use:n</code>	104, 157, 169, 237, 446
<code>\use_i:nnn</code>	307
<code>\use_ii:nnn</code>	307

<code>\use_iii:nnn</code>	293
<code>\use_none:nn</code> ..	87, 88, 89, 90, 91, 92, 93
<code>\UTFencname</code>	49, 82

X	
<code>\XeLaTeX</code>	33
<code>\XeTeXcountvariations</code>	355
<code>\XeTeXfeaturename</code>	28
<code>\XeTeXfonttype</code>	348
<code>\XeTeXisexclusivefeature</code>	32
<code>\XeTeXOTcountfeatures</code>	192
<code>\XeTeXOTcountlanguages</code>	138
<code>\XeTeXOTcountscripts</code>	87
<code>\XeTeXOTfeaturetag</code>	198
<code>\XeTeXOTlanguagetag</code>	144
<code>\XeTeXOTscripttag</code>	92
<code>\XeTeXpicfile</code>	60, 61, 63
<code>\XeTeXselectorname</code>	34, 39, 44
<code>\Xi</code>	56