

# The fontspec package

## Font selection for X<sub>Y</sub>LaTeX and LuaLaTeX

WILL ROBERTSON

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

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

2018/07/30      vv2.6h

## Contents

<b>I</b>	<b>fontspec.dtx</b>	<b>6</b>
1	Package declaration	6
1.1	Lua header . . . . .	7
<b>II</b>	<b>fontspec-code-load.dtx</b>	<b>8</b>
1	The fontspec.sty loading file	8
<b>III</b>	<b>fontspec-vars.dtx</b>	<b>9</b>
1	Declaration of variables	9
<b>IV</b>	<b>fontspec-msg.dtx</b>	<b>13</b>
1	Error/warning/info messages	13
1.1	Errors . . . . .	13
1.2	Warnings . . . . .	14
1.3	Info messages . . . . .	16
<b>V</b>	<b>fontspec-opening.dtx</b>	<b>18</b>
1	Opening code	18
1.1	Package options . . . . .	18
1.2	Encodings . . . . .	18

1.3	Generic functions	19
1.4	expl3 variants	20
<b>VI</b>	<b>fontspec-fontload.dtx</b>	<b>21</b>
1	expl3 interface for primitive font loading	21
<b>VII</b>	<b>fontspec-interfaces.dtx</b>	<b>23</b>
1	User commands	23
<b>VIII</b>	<b>fontspec-user.dtx</b>	<b>26</b>
1	User command internals	26
1.1	Font selection	26
1.2	Font feature selection	29
1.3	Defining new font features	31
<b>IX</b>	<b>fontspec-api.dtx</b>	<b>34</b>
1	Programmer's interface	34
<b>X</b>	<b>fontspec-internal.dtx</b>	<b>40</b>
1	Internals	40
1.1	The main function for setting fonts	40
1.2	Setting font shapes in a family	48
1.3	Initialisation	57
1.4	Miscellaneous	57
<b>XI</b>	<b>fontspec-opentype.dtx</b>	<b>59</b>
1	OpenType definitions code	59
1.1	Adding features when loading fonts	60
1.2	OpenType feature information	63
<b>XII</b>	<b>fontspec-graphite.dtx</b>	<b>67</b>
1	Graphite/AAT code	67
<b>XIII</b>	<b>fontspec-keyval.dtx</b>	<b>69</b>
1	Font loading (keyval) definitions	69

1.1	Pre-pre-parsing stages . . . . .	69
1.2	Pre-parsed features . . . . .	71
1.3	Font faces . . . . .	71
1.4	General font-independent features . . . . .	74
<b>XIV fontspec-feat-opentype.dtx</b>		<b>84</b>
1	OpenType feature definitions	84
2	Regular key=val / tag definitions	84
2.1	Ligatures . . . . .	84
2.2	Letters . . . . .	84
2.3	Numbers . . . . .	85
2.4	Vertical position . . . . .	85
2.5	Contextuals . . . . .	85
2.6	Diacritics . . . . .	86
2.7	Kerning . . . . .	86
2.8	Fractions . . . . .	86
2.9	Style . . . . .	87
2.10	CJK shape . . . . .	87
2.11	Character width . . . . .	88
2.12	Vertical . . . . .	88
3	OpenType features that need numbering	88
3.1	Alternate . . . . .	88
3.2	Variant / StylisticSet . . . . .	89
3.3	CharacterVariant . . . . .	89
3.4	Annotation . . . . .	89
3.5	Ornament . . . . .	90
4	Script and Language	90
4.1	Script . . . . .	90
4.2	Language . . . . .	91
5	Backwards compatibility	92
<b>XV fontspec-scripts.dtx</b>		<b>93</b>
1	Font script definitions	93
<b>XVI fontspec-lang.dtx</b>		<b>96</b>
1	Font language definitions	96

<b>XVII</b>	<b>fontspec-feat-aat.dtx</b>	<b>104</b>
1	AAT feature definitions	104
1.1	Ligatures	104
1.2	Letters	104
1.3	Numbers	105
1.4	Contextuals	105
1.5	Diacritics	105
1.6	Vertical position	105
1.7	Fractions	105
1.8	Alternate	105
1.9	Variant / StylisticSet	106
1.10	Style	106
1.11	CJK shape	106
1.12	Character width	107
1.13	Annotation	107
<b>XVIII</b>	<b>fontspec-enc.dtx</b>	<b>108</b>
1	Extended font encodings	108
<b>XIX</b>	<b>fontspec-math.dtx</b>	<b>111</b>
1	Selecting maths fonts	111
<b>XX</b>	<b>fontspec-closing.dtx</b>	<b>116</b>
1	Closing code	116
1.1	Finishing up	116
<b>XXI</b>	<b>fontspec-xfss.dtx</b>	<b>117</b>
1	Changes to the NFSS	117
1.1	Italic small caps and so on	117
1.2	Emphasis	118
1.3	Strong emphasis	120
<b>XXII</b>	<b>fontspec-patches.dtx</b>	<b>122</b>
1	Patching code	122
1.1	\-	122
1.2	Verbatims	122
1.3	\oldstylenums	124



# 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-vars.dtx}
6   \DTX{fontspec-msg.dtx}
7   \DTX{fontspec-opening.dtx}
8   \DTX{fontspec-fontload.dtx}
9   \DTX{fontspec-interfaces.dtx}
10  \DTX{fontspec-user.dtx}
11  \DTX{fontspec-api.dtx}
12  \DTX{fontspec-internal.dtx}
13  \DTX{fontspec-opentype.dtx}
14  \DTX{fontspec-graphite.dtx}
15  \DTX{fontspec-keyval.dtx}
16  \DTX{fontspec-feat-opentype.dtx}
17  \DTX{fontspec-scripts.dtx}
18  \DTX{fontspec-lang.dtx}
19  \DTX{fontspec-feat-aat.dtx}
20  \DTX{fontspec-enc.dtx}
21  \DTX{fontspec-math.dtx}
22  \DTX{fontspec-closing.dtx}
23  \DTX{fontspec-xfss.dtx}
24  \DTX{fontspec-patches.dtx}
25 }
26 </dtx>
```

Now exit if we're using plain T<sub>E</sub>X; this would usually be the case when loading this file with fontspec.ins.

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

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

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

```

40 \url{http://wspr.io/fontspec/}
41 }
42 </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.

```

43 <fontspec>\RequirePackage{xparse}
44 <fontspec & load>\ProvidesExplPackage{fontspec}%
45 <fontspec & XE>\ProvidesExplPackage{fontspec-xetex}%
46 <fontspec & LU>\ProvidesExplPackage{fontspec-luatex}%
47 <*dtx>
48 \RequirePackage{xparse}
49 \ProvidesExplFile{fontspec.dtx}
50 </dtx>
51 <*fontspec>
52 {2018/07/30}{v2.6h}{Font selection for XeLaTeX and LuaLaTeX}
53 </fontspec>

```

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

```

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

```

## 1.1 Lua header

```

58 <lua>fontspec = fontspec or {}
59 <lua>local fontspec = fontspec
60 <lua>fontspec.module = {
61 <lua>    name = "fontspec",
62 <lua>    version = "v2.6h",
63 <lua>    date = "2018/07/30",
64 <lua>    description = "Font selection for XeLaTeX and LuaLaTeX",
65 <lua>    author = "Khaled Hosny, Philipp Gesang, Will Robertson",
66 <lua>    copyright = "Khaled Hosny, Philipp Gesang, Will Robertson",
67 <lua>    license = "LPPL v1.3c"
68 <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<sup>®</sup>TeX

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

### X<sub>Ǝ</sub>TeX

```
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-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_@@_graphite_bool
11 \bool_new:N \l_@@_fontcfg_bool
12 \bool_set_true:N \l_@@_fontcfg_bool
```

For dealing with legacy maths:

```
13 \bool_new:N \g_@@_math_euler_bool
14 \bool_new:N \g_@@_math_lucida_bool
15 \bool_new:N \g_@@_pkg_euler_loaded_bool
```

For package options:

```
16 \bool_new:N \g_@@_cfg_bool
17 \bool_new:N \g_@@_math_bool
18 \bool_new:N \g_@@_euenc_bool

19 \bool_new:N \l_@@_tmpa_bool
20 \bool_new:N \l_@@_disable_defaults_bool
21 \bool_new:N \l_@@_alias_bool
22 \bool_new:N \l_@@_external_bool
23 \bool_new:N \l_@@_never_check_bool
24 \bool_new:N \l_@@_defining_encoding_bool
25 \bool_new:N \l_@@_script_exist_bool
26 \bool_new:N \g_@@_em_normalise_slant_bool
27 \bool_new:N \l_@@_proceed_bool
28 \bool_new:N \l_@@_check_feat_bool
```

## 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_@@_em_int
34 \int_new:N \l_@@_emdef_int
35 \int_new:N \l_@@_strong_int
36 \int_new:N \l_@@_strongdef_int
```

## Floats

```
37 \fp_new:N \l_@@_tmpa_fp
38 \fp_new:N \l_@@_tmpb_fp
```

## Dimensions

```
39 \dim_new:N \l_@@_tmpa_dim
40 \dim_new:N \l_@@_tmpb_dim
41 \dim_new:N \l_@@_tmpc_dim
```

## Sequences

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

## Comma-lists

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

58 \clist_new:N \l_@@_fontfeat_up_clist
59 \clist_new:N \l_@@_fontfeat_bf_clist
60 \clist_new:N \l_@@_fontfeat_it_clist
61 \clist_new:N \l_@@_fontfeat_bfit_clist
62 \clist_new:N \l_@@_fontfeat_sl_clist
63 \clist_new:N \l_@@_fontfeat_bfsl_clist
64 \clist_new:N \l_@@_fontfeat_sc_clist
```

## Property lists

```
65 \prop_new:N \g_@@_fontopts_prop
66 \prop_new:N \l_@@_nfss_prop
67 \prop_new:N \l_@@_nfssfont_prop
68 \prop_new:N \g_@@_OT_features_prop
69 \prop_new:N \g_@@_all_opentype_feature_names_prop
70 \prop_new:N \g_@@_em_prop
71 \prop_new:N \g_@@_strong_prop
72 \prop_new:N \g_@@_fontid_family_prop
73 \prop_new:N \g_@@_family_int_prop
```

## Token lists

```
74 \tl_new:N \l_fontspec_family_tl
75 \tl_new:N \g_fontspec_encoding_tl
76 \tl_new:N \l_fontspec_renderer_tl
77 \tl_new:N \l_fontspec_fontname_tl

78 \tl_clear_new:N \UTFencname
79 \tl_clear_new:N \cyrillicencoding
80 \tl_clear_new:N \latinencoding

81 \tl_new:N \l_fontspec_mode_tl
82 \tl_new:N \g_@@_curr_series_tl
83 \tl_new:N \g_@@_defined_shapes_tl
84 \tl_new:N \g_@@_nfss_enc_tl
85 \tl_new:N \g_@@_nfss_family_tl
86 \tl_new:N \g_@@_single_feat_tl
87 \tl_new:N \l_@@_basename_tl
88 \tl_new:N \l_@@_curr_fontname_tl
89 \tl_new:N \l_@@_curr_bfname_tl
90 \tl_new:N \l_@@_ext_filename_tl
91 \tl_new:N \l_@@_extension_tl
92 \tl_new:N \l_@@_font_path_tl
93 \tl_new:N \l_@@_fontid_tl
94 \tl_new:N \l_@@_fontname_tl
95 \tl_new:N \l_@@_hexcol_tl
96 \tl_new:N \l_@@_nfss_sc_tl
97 \tl_new:N \l_@@_nfss_tl
98 \tl_new:N \l_@@_nfss_fam_tl
99 \tl_new:N \l_@@_opacity_tl
100 \tl_new:N \l_@@_optical_size_tl
101 \tl_new:N \l_@@_options_tl
102 \tl_new:N \l_@@_saved_fontname_tl
103 \tl_new:N \l_@@_scale_tl
104 \tl_new:N \l_@@_size_tl
105 \tl_new:N \l_@@_sizedfont_tl
106 \tl_new:N \l_@@_this_font_tl
107 \tl_new:N \l_@@_tmp_tl
108 \tl_new:N \l_@@_tmpa_tl
109 \tl_new:N \l_@@_tmpb_tl
110 \tl_new:N \l_@@_ttc_index_tl
111 \tl_new:N \l_@@_emshape_query_tl
```

```

112 \tl_new:N \l_@@_em_switch_tl
113 \tl_new:N \l_@@_em_tmp_tl

114 \tl_new:N \g_@@_mathrm_tl
115 \tl_new:N \g_@@_bfmathrm_tl
116 \tl_new:N \g_@@_mathsf_tl
117 \tl_new:N \g_@@_mathtt_tl

    Defaults:

118 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
119 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
120 \tl_gset:Nn \g_@@_mathtt_tl {\ttdefault}

121 \tl_new:N \l_@@_family_label_tl
122 \tl_new:N \l_@@_fake_slant_tl
123 \tl_new:N \l_@@_fake_embolden_tl

124 \tl_new:N \l_@@_fontname_up_tl
125 \tl_new:N \l_@@_fontname_bf_tl
126 \tl_new:N \l_@@_fontname_it_tl
127 \tl_new:N \l_@@_fontname_bfit_tl
128 \tl_new:N \l_@@_fontname_sl_tl
129 \tl_new:N \l_@@_fontname_bfsl_tl
130 \tl_new:N \l_@@_fontname_sc_tl

131 \tl_new:N \l_@@_script_name_tl
132 \tl_new:N \l_fontspect_script_tl
133 \tl_new:N \l_@@_lang_name_tl
134 \tl_new:N \l_fontspect_lang_tl

135 \tl_new:N \l_@@_mapping_tl
136 \tl_new:N \l_@@_punctspace_adjust_tl
137 \tl_new:N \l_@@_wordspace_adjust_tl
138 \tl_new:N \l_@@_postadjust_tl

139 \tl_const:Nn \c_@@_hexcol_tl {000000}
140 \tl_const:Nn \c_@@_opacity_tl {FF~}
141 \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.

```

142 \tl_new:N \g_@@_rawfeatures_sclist
143 \tl_new:N \l_@@_pre_feat_sclist

```

**Font families** Again not a real data structure, and also probably poorly named.

```

144 \tl_new:N \l_@@_rmfamily_family_tl
145 \tl_new:N \l_@@_sffamily_family_tl
146 \tl_new:N \l_@@_ttfamily_family_tl

```

## File IV

# fontspec-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:nnn
14 { \msg_new:nnx {#1} {#2} { \tl_trim_spaces:n {#3} } }
15 \cs_new:Nn \@@_msg_new:nnnn
16 { \msg_new:nnxx {#1} {#2} { \tl_trim_spaces:n {#3} } { \tl_trim_spaces:n {#4} } }
17 \char_set_catcode_space:n {32}
```

### 1.1 Errors

```
18 \@@_msg_new:nnn {fontspec} {only-inside-encdef}
19 {
20   \exp_not:N#1can only be used in the second argument
21   to \string\DeclareUnicodeEncoding.
22 }
23 \@@_msg_new:nnn {fontspec} {no-size-info}
24 {
25   Size information must be supplied.\\
26   For example, SizeFeatures={Size={8-12},...}.
27 }
28 \@@_msg_new:nnnn {fontspec} {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:nnnn {fontspec} {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:nnn {fontspec} {no-glyph}
45 {
46   '\l_fontspec_fontname_tl' does not contain glyph #1.
47 }
48 \@@_msg_new:nnnn {fontspec} {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:nnnn {fontspec} {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:nnnn {fontspec} {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:nnnn {fontspec} {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:nnn {fontspec} {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:nnn {fontspec} {tu-missing}
91   {
92     The TU encoding seems to be missing; please update to the latest version of LaTeX2e.
93   }
94   \@@_msg_new:nnn {fontspec} {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:nnn {fontspec} {feature-option-overwrite}
105    {
106      Option '#2' of font feature '#1' overwritten.
107    }
108    \@@_msg_new:nnn {fontspec} {script-not-exist-latn}
109    {
110      Font '\l_fontspec_fontname_tl' does not contain script '#1'.\\
111      'Latin' script used instead.
112    }
113    \@@_msg_new:nnn {fontspec} {script-not-exist}
114    {
115      Font '\l_fontspec_fontname_tl' does not contain script '#1'.
116    }
117    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist}
118    {
119      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
120      for AAT font '\l_fontspec_fontname_tl'.
121    }
122    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist-in-font}
123    {
124      AAT feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
125      in font '\l_fontspec_fontname_tl'.
126    }
127    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist}
128    {
129      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
130      for OpenType font '\l_fontspec_fontname_tl'
131    }
132    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist-in-font}
133    {
134      OpenType feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
135      for font '\l_fontspec_fontname_tl'
136      with script '\l_@@_script_name_tl' and language '\l_@@_lang_name_tl'.
137    }
138    \@@_msg_new:nnn {fontspec} {no-opticals}

```

```

139 {
140   '\l_fontspec_fontname_tl' doesn't appear to have an Optical Size axis.
141 }
142 \@@_msg_new:nnn {fontspec} {language-not-exist}
143 {
144   Language '#1' not available
145   for font '\l_fontspec_fontname_tl'
146   with script '\l_@@_script_name_tl'.\\
147   'Default' language used instead.
148 }
149 \@@_msg_new:nnn {fontspec} {only-xetex-feature}
150 {
151   Ignored XeTeX only feature: '#1'.
152 }
153 \@@_msg_new:nnn {fontspec} {only-luatex-feature}
154 {
155   Ignored LuaTeX only feature: '#1'.
156 }
157 \@@_msg_new:nnn {fontspec} {no-mapping}
158 {
159   Input mapping not (yet?) supported in LuaTeX.
160 }
161 \@@_msg_new:nnn {fontspec} {no-mapping-ligtx}
162 {
163   Input mapping not (yet?) supported in LuaTeX.\\
164   Use "Ligatures=TeX" instead of "Mapping=tex-text".
165 }
166 \@@_msg_new:nnn {fontspec} {cm-default-obsolete}
167 {
168   The "cm-default" package option is obsolete.
169 }
170 \@@_msg_new:nnn {fontspec} {fakebold-only-xetex}
171 {
172   The "FakeBold" and "AutoFakeBold" options are only available with XeLaTeX.\\
173   Option ignored.
174 }
175 \@@_msg_new:nnn {fontspec} {font-index-needs-ttc}
176 {
177   The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
178   Feature ignored.
179 }
180 \@@_msg_new:nnn {fontspec} {feat-cannot-remove}
181 {
182   The "#1" feature cannot be deactivated. Request ignored.
183 }

```

### 1.3 Info messages

```

184 \@@_msg_new:nnn {fontspec} {defining-font}
185 {
186   Font family '\g_@@_nfss_family_tl' created for font '#2'
187   with options [\l_@@_all_features_clist].\\

```

```

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

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

```

## File V

# fontspec-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 \DeclareOption{silent}
19 {
20   \msg_redirect_module:nnn { fontspec } { warning } { none }
21   \msg_redirect_module:nnn { fontspec } { info } { none }
22 }
23
24 \ExecuteOptions{config,math,tuenc}
25 \ProcessOptions*
```

### 1.2 Encodings

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

```
23 \bool_if:NF \g_@@_euenc_bool
24 {
25   \file_if_exist:nTF {tuenc.def}
26   {
27     \cs_if_exist:cF {T@TU}
28     {
29       \@@_warning:n {tu-clash}
30       \DeclareFontEncoding{TU}{}{}
31       \DeclareFontSubstitution{TU}{lmr}{m}{n}
32     }
33   }
34   {
35     \@@_warning:n {tu-missing}
36     \bool_gset_true:N \g_@@_euenc_bool
37   }
38 }
```

```

39 \bool_if:NTF \g_@@_euenc_bool
40 {
41   \tl_gset:Nn \g_fontspec_encoding_tl {EU1}
42   \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`  
`\FontspecSetCheckBoolFalse`

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

```

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 ??.)

## 1.4 expl3 variants

```

78 \cs_generate_variant:Nn \int_set:Nn {Nv}
79 \cs_generate_variant:Nn \keys_set:nn {nx}
80 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
81 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
82 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
83 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
84 \cs_generate_variant:Nn \prop_gput:Nnn {NxN}
85 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
86 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
87 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
88 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
89 \cs_generate_variant:Nn \tl_if_empty:nF {x}
90 \cs_generate_variant:Nn \tl_if_empty:nF {f}
91 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
92 \cs_generate_variant:Nn \tl_replace_all:Nnn {Nnx}

```

## File VI

# fontspec-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~ #3 \scan_stop:
4 }
5 \cs_set:Npn \@@_primitive_font_gset:Nnn #1#2#3
6 {
7   \global \font #1 = #2 ~at~ #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_eq:NN \xetex_suppressfontnotfounderror:D \c_one
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_if_exist:nTF
21 \prg_set_conditional:Nnn \@@_primitive_font_if_exist:n {TF,T,F}
22 {
23   \group_begin:
24     \@@_font_suppress_not_found_error:
25     \@@_primitive_font_set:Nnn \l_@@_primitive_font {#1} {10pt}
26     \@@_primitive_font_if_null:NTF \l_@@_primitive_font
27     { \group_end: \prg_return_false: }
28     { \group_end: \prg_return_true: }
29 }
```

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

`\@@_primitive_font_glyph_if_exist:NnTF`

```
30 \prg_new_conditional:Nnn \@@_primitive_font_glyph_if_exist:Nn {p,TF,T,F}  
31 {  
32   \etex_iffontchar:D #1 #2 \scan_stop:  
33   \prg_return_true:  
34   \else:  
35   \prg_return_false:  
36   \fi:  
37 }
```

*(End definition for \@@\_primitive\_font\_glyph\_if\_exist:NnTF. This function is documented on page ??.)*

`\@@_primitive_font_set_hyphenchar:Nn`

```
38 \cs_new:Nn \@@_primitive_font_set_hyphenchar:Nn  
39 {  
40   \tex_hyphenchar:D #1 = #2 \scan_stop:  
41 }
```

*(End definition for \@@\_primitive\_font\_set\_hyphenchar:Nn. This function is documented on page ??.)*

## File VII

# fontspec-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 26](#).

```
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 \newfontface { m O{} m O{} }
54 {
55     \@@_main_newfontface:nnn {#1} {#2,#4} {#3}
56 }

```

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

```

57 \NewDocumentCommand \defaultfontfeatures { t+ o m }
58 {
59     \IfNoValueTF {#2}
60     { \@@_set_default_features:nn {#1} {#3} }
61     { \@@_set_font_default_features:nnn {#1} {#2} {#3} }
62     \ignorespaces
63 }

```

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

```

64 \NewDocumentCommand \addfontfeatures {m}
65 {
66     \@@_main_addfontfeatures:n {#1}
67 }
68 \NewDocumentCommand \addfontfeature {m}
69 {
70     \@@_main_addfontfeatures:n {#1}
71 }
72 \NewDocumentCommand \newfontfeature {mm}
73 {
74     \@@_main_newfontfeature:nn {#1} {#2}
75 }

```

```

76 \NewDocumentCommand \newAATfeature {mmm}
77 {
78   \@@_main_newAATfeature:nnnn {#1} {#2} {#3} {#4}
79 }

80 \NewDocumentCommand \newopentypefeature {mmm}
81 {
82   \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
83 }

```

`\newICUfeature` Deprecated.

```

84 \NewDocumentCommand \newICUfeature {mmm}
85 {
86   \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
87 }

```

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

```

88 \NewDocumentCommand \aliasfontfeature {mm}
89 {
90   \@@_main_aliasfontfeature:nn {#1} {#2}
91 }

92 \NewDocumentCommand \aliasfontfeatureoption {mmm}
93 {
94   \@@_main_aliasfontfeatureoption:nnn {#1} {#2} {#3}
95 }

```

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

```

96 \NewDocumentCommand \newfontscript {mm}
97 {
98   \fontspec_new_script:nn {#1} {#2}
99 }

```

*(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.

```

100 \NewDocumentCommand \newfontlanguage {mm}
101 {
102   \fontspec_new_lang:nn {#1} {#2}
103 }

```

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

```

104 \NewDocumentCommand \DeclareFontExtensions {m}
105 {
106   \@@_main_DeclareFontExtensions:n {#1}
107 }

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

```

## File VIII

# fontspec-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 ??.)*

`\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.

```
7 \cs_new:Nn \@@_main_setmainfont:nn
8 {
9   \fontspec_set_family:Nnn \l_@@_rmfamily_family_tl {#1} {#2}
10  \tl_set_eq:NN \rmdefault \l_@@_rmfamily_family_tl
11  \use:x
12  {
13    \exp_not:n { \DeclareRobustCommand \rmfamily }
14    {
15      \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
16      \exp_not:N \fontfamily { \l_@@_rmfamily_family_tl }
17      \exp_not:N \selectfont
18    }
19  }
20  \str_if_eq:x:nnT {\familydefault} {\rmdefault}
21  { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
22  \@@_setmainfont_hook:nn {#1} {#2}
23  \normalfont
24 }
```

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

`\setsansfont` Same as above.

```
25 \cs_new:Nn \@@_main_setsansfont:nn
26 {
27   \fontspec_set_family:Nnn \l_@@_sffamily_family_tl {#1} {#2}
28   \tl_set_eq:NN \sfdefault \l_@@_sffamily_family_tl
29   \use:x
30   {
```

```

31     \exp_not:n { \DeclareRobustCommand \sffamily }
32     {
33         \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
34         \exp_not:N \fontfamily { \l_@@_sffamily_family_tl }
35         \exp_not:N \selectfont
36     }
37 }
38 \str_if_eq_x:nnT {\familydefault} {\sfdefault}
39 { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
40 \@@_setsansfont_hook:nn {#1} {#2}
41 \normalfont
42 }

```

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

`\setmonofont` Same as above.

```

43 \cs_new:Nn \@@_main_setmonofont:nn
44 {
45     \fontspec_set_family:Nnn \l_@@_ttfamily_family_tl {#1} {#2}
46     \tl_set_eq:NN \ttdefault \l_@@_ttfamily_family_tl
47     \use:x
48     {
49         \exp_not:n { \DeclareRobustCommand \ttfamily }
50         {
51             \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
52             \exp_not:N \fontfamily { \l_@@_ttfamily_family_tl }
53             \exp_not:N \selectfont
54         }
55     }
56     \str_if_eq_x:nnT {\familydefault} {\ttdefault}
57     { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
58     \@@_setmonofont_hook:nn {#1} {#2}
59     \normalfont
60 }

```

(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`.

```

61 \cs_new:Nn \@@_main_setmathrm:nn
62 {
63     \fontspec_set_family:Nnn \g_@@_mathrm_tl {#1} {#2}
64     \fontspec_set_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
65     \@@_setmathrm_hook:nn {#1} {#2}
66 }

```

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

`\setboldmathrm`

```

67 \cs_new:Nn \@@_main_setboldmathrm:nn
68 {

```

```

69 <XE> \fontspec_set_family:Nnn \g_@@_bfmathrm_tl {#1} {#2}
70 <LU> \fontspec_set_family:Nnn \g_@@_bfmathrm_tl {Renderer=Basic,#1} {#2}
71     \@@_setboldmathrm_hook:nn {#1} {#2}
72 }

```

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

`\setmathsf`

```

73 \cs_new:Nn \@@_main_setmathsf:nn
74 {
75 <XE> \fontspec_set_family:Nnn \g_@@_mathsf_tl {#1} {#2}
76 <LU> \fontspec_set_family:Nnn \g_@@_mathsf_tl {Renderer=Basic,#1} {#2}
77     \@@_setmathsf_hook:nn {#1} {#2}
78 }

```

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

`\setmathtt`

```

79 \cs_new:Nn \@@_main_setmathtt:nn
80 {
81 <XE> \fontspec_set_family:Nnn \g_@@_mathtt_tl {#1} {#2}
82 <LU> \fontspec_set_family:Nnn \g_@@_mathtt_tl {Renderer=Basic,#1} {#2}
83     \@@_setmathtt_hook:nn {#1} {#2}
84 }

```

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

Hooks:

```

85 \cs_set_eq:NN \@@_setmainfont_hook:nn \use_none:nn
86 \cs_set_eq:NN \@@_setsansfont_hook:nn \use_none:nn
87 \cs_set_eq:NN \@@_setmonofont_hook:nn \use_none:nn
88 \cs_set_eq:NN \@@_setmathrm_hook:nn \use_none:nn
89 \cs_set_eq:NN \@@_setmathsf_hook:nn \use_none:nn
90 \cs_set_eq:NN \@@_setmathtt_hook:nn \use_none:nn
91 \cs_set_eq:NN \@@_setboldmathrm_hook:nn \use_none:nn

```

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

```

92 \onlypreamble\setmathrm
93 \onlypreamble\setboldmathrm
94 \onlypreamble\setmathsf
95 \onlypreamble\setmathtt

```

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

```

96 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
97 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
98 \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).

```

99 \cs_new:Nn \@@_main_newfontfamily:nnnN
100 {
101     \fontspec_set_family:cnn { l_@@_ \cs_to_str:N #1 _family_tl } {#2} {#3}
102     \use:x

```

```

103 {
104   \exp_not:N #4 \exp_not:N #1 {}
105   {
106     \exp_not:N \fontfamily { \use:c { l_@@_ \cs_to_str:N #1 _family_tl } }
107     \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
108     \exp_not:N \selectfont
109   }
110 }
111 }

```

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

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

```

112 \cs_new:Nn \@@_main_newfontface:nnn
113 {
114   \newfontfamily #1 [ BoldFont={},ItalicFont={},SmallCapsFont={},#2 ] {#3}
115 }

```

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

## 1.2 Font feature selection

\@@\_set\_default\_features:nn

```

116 \cs_new:Nn \@@_set_default_features:nn
117 {
118   \IfBooleanTF {#1} \clist_gput_right:Nn \clist_gset:Nn
119     \g_@@_default_fontopts_clist {#2}
120 }

```

(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.

```

121 \cs_new:Nn \@@_set_font_default_features:nnn
122 {
123   <debug> \typeout{\unexpanded{\set_font_default_features:nnn:{#1}{#2}{#3}}}
124   \clist_map_inline:nn {#2}
125   {
126     \tl_if_single:nTF {##1}
127     { \tl_set:No \l_@@_tmp_tl { \cs:w l_@@_ \cs_to_str:N ##1 _family_tl\cs_end: } }
128     { \@@_sanitise_fontname:Nn \l_@@_tmp_tl {##1} }
129
130     \IfBooleanTF {#1}
131     {
132       \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
133       { \tl_clear:N \l_@@_tmpb_tl }
134       \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
135       \prop_gput:NVV \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
136     }
137   }

```

```

138         \tl_if_empty:nTF {#3}
139         { \prop_gremove:NV \g_@@_fontopts_prop \l_@@_tmp_tl }
140         { \prop_gput:NVN \g_@@_fontopts_prop \l_@@_tmp_tl {#3,} }
141     }
142 }
143 }

```

(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.

```

144 \cs_new:Nn \@@_main_addfontfeatures:n
145 {
146   <debug> \typeout{^^J::::::::::::::::::::::::::::::::::::^^J: addfontfeatures}
147   \fontspec_if_fontspec_font:TF
148   {
149     \group_begin:
150     \keys_set_known:nnN {fontspec-addfeatures} {#1} \l_@@_tmp_tl
151     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {options} \l_@@_options_tl
152     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {fontname} \l_@@_fontname_tl
153     \bool_set_true:N \l_@@_disable_defaults_bool
154     <debug> \typeout{ \@@_select_font_family:nn { \l_@@_options_tl , #1 } {\l_@@_fontname_tl} }
155     \use:x
156     {
157       \@@_select_font_family:nn
158       { \l_@@_options_tl , #1 } {\l_@@_fontname_tl}
159     }
160     \group_end:
161     \fontfamily \g_@@_nfss_family_tl \selectfont
162   }
163   {
164     \@@_warning:nx {addfontfeatures-ignored} {#1}
165   }
166   \ignorespaces
167 }

```

(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.

```
168 \cs_new:Nn \@@_main_newfontfeature:nn
169 {
170   \keys_define:nn { fontspec }
171   {
172     #1 .code:n = { \@@_update_featstr:n {#2} }
173   }
174 }
```

*(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.

```
175 \cs_new:Nn \@@_main_newAATfeature:nnnn
176 {
177   \keys_if_exist:nnF { fontspec } {#1}
178   { \@@_define_aat_feature_group:n {#1} }
179
180   \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
181   { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
182
183   \@@_define_aat_feature:nnnn {#1}{#2}{#3}{#4}
184 }
```

*(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.

```
185 \cs_new:Nn \@@_main_newopentypefeature:nnn
186 {
187   \keys_if_exist:nnF { fontspec / options } {#1}
188   { \@@_define_opentype_feature_group:n {#1} }
189
190   \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
191   { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
192
193   \exp_args:Nnnx \@@_define_opentype_feature:nnnnn
194   {#1} {#2} { \@@_strip_plus_minus:n {#3} } {#3} {}
195 }
196 \cs_new:Nn \@@_strip_plus_minus:n { \@@_strip_plus_minus_aux:Nq #1 \q_nil }
197 \cs_new:Npn \@@_strip_plus_minus_aux:Nq #1#2 \q_nil
198 {
199   \str_case:nnF {#1} { {+} {#2} {-} {#2} } {#1#2}
200 }
```

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

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

```

201 \cs_new:Nn \@@_main_aliasfontfeature:nn
202 {
203   <debug> \typeout{::::::::::::::::::::~^J:: aliasfontfeature{#1}{#2}}
204   \bool_set_false:N \l_@@_alias_bool
205
206   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
207   {
208     \keys_if_exist:nnT {##1} {#1}
209     {
210       <debug> \typeout{::: Key~exists~##1~/~#1}
211       \bool_set_true:N \l_@@_alias_bool
212       \keys_define:nn {##1}
213       { #2 .code:n = { \keys_set:nn {##1} { #1 = {####1} } } }
214     }
215   }
216
217   \bool_if:NF \l_@@_alias_bool
218   { \@@_warning:nx {rename-feature-not-exist} {#1} }
219 }

```

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

`\aliasfontfeatureoption`

```

220 \cs_new:Nn \@@_main_aliasfontfeatureoption:nnn
221 {
222   \bool_set_false:N \l_@@_alias_bool
223
224   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
225   {
226     \keys_if_exist:nnT { ##1 / #1 } {#2}
227     {
228       <debug> \typeout{::: Keyval~exists~##1~/~#1~=#2}
229       \bool_set_true:N \l_@@_alias_bool
230       \keys_define:nn { ##1 / #1 }
231       { #3 .code:n = { \keys_set:nn {##1} { #1 = {#2} } } }
232     }
233
234     \keys_if_exist:nnT { ##1 / #1 } {#2Reset}
235     {
236       <debug> \typeout{::: Keyval~exists~##1~/~#1~=#2Reset}
237       \keys_define:nn { ##1 / #1 }
238       { #3Reset .code:n = { \keys_set:nn {##1} { #1 = {#2Reset} } } }
239     }
240
241     \keys_if_exist:nnT { ##1 / #1 } {#20ff}
242     {
243       <debug> \typeout{::: Keyval~exists~##1~/~#1~=#20ff}
244       \keys_define:nn { ##1 / #1 }
245       { #30ff .code:n = { \keys_set:nn {##1} { #1 = {#20ff} } } }
246     }
247   }

```

```

248
249 \bool_if:NF \l_@@_alias_bool
250 { \@@_warning:nx {rename-feature-not-exist} {#1/#2} }
251 }

```

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

\@@\_main\_DeclareFontExtensions:n

```

252 \cs_new:Nn \@@_main_DeclareFontExtensions:n
253 {
254   \clist_set:Nn \l_@@_extensions_clist { #1 }
255 }

```

Defaults:

```

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

```

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

\IfFontFeatureActiveTF

```

257 \cs_new:Nn \@@_main_IfFontFeatureActiveTF:nnn
258 {
259   <debug> \typeout{^^J::::::::::::::::::::::::::::::::::::::::::::::::::}
260   <debug> \typeout{:IfFontFeatureActiveTF \exp_not:n{#{1}{#2}{#3}}{}}
261   \@@_if_font_feature:nTF {#1} {#2} {#3}
262 }
263 \prg_new_conditional:Nnn \@@_if_font_feature:n {TF}
264 {
265   \tl_gclear:N \g_@@_single_feat_tl
266   \group_begin:
267     \@@_font_suppress_not_found_error:
268     \@@_init:
269     \bool_set_true:N \l_@@_ot_bool
270     \bool_set_true:N \l_@@_never_check_bool
271     \bool_set_false:N \l_@@_firsttime_bool
272     \clist_clear:N \l_@@_fontfeat_clist
273     \@@_get_features:n {#1}
274   \group_end:
275
276   <debug> \typeout{:::> \exp_not:N\g_@@_rawfeatures_sclist->~{\g_@@_rawfeatures_sclist}}
277   <debug> \typeout{:::> \exp_not:N\g_@@_single_feat_tl->~{\g_@@_single_feat_tl}}
278
279   \tl_if_empty:NTF \g_@@_single_feat_tl { \prg_return_false: }
280   {
281     \exp_args:NV \fontspec_if_current_feature:nTF \g_@@_single_feat_tl
282     { \prg_return_true: } { \prg_return_false: }
283   }
284 }

```

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

## File IX

# fontspec-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:  
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_fontspec_script_t
48       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag} \l_fontspec_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     \l_@@_script_int {#1}
68     \l_@@_language_int {#2}
69     \font {#3} \prg_return_true: \prg_return_false:
70   }
71   { \prg_return_false: }
72 }
73 { \prg_return_false: }
74 }

```

(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.

```

75 \prg_new_conditional:Nnn \fontspec_if_script:n {TF,T,F}
76 {
77   \fontspec_if_fontspec_font:TF
78   {
79     \set_font_type:N \font
80     \bool_if:NTF \l_@@_ot_bool
81     {
82       \font {#1} \prg_return_true: \prg_return_false:
83     }
84     { \prg_return_false: }
85   }
86   { \prg_return_false: }
87 }

```

(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.

```

88 \prg_new_conditional:Nnn \fontspec_if_language:n {TF,T,F}
89 {
90   \fontspec_if_fontspec_font:TF
91   {
92     \set_font_type:N \font
93     \bool_if:NTF \l_@@_ot_bool
94     {
95       \prop_get:cnN {g_@@_fontinfo_ \fontfamily_prop} {script-num} \l_@@_tmp_tl
96       \int_set:Nn \l_@@_script_int {\l_@@_tmp_tl}
97       \prop_get:cnN {g_@@_fontinfo_ \fontfamily_prop} {script-tag} \l_fontspec_script_tl
98       \font {#1} \prg_return_true: \prg_return_false:
99     }
100     { \prg_return_false: }
101   }
102   { \prg_return_false: }
103 }
104 }

```

(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.

```

105 \prg_new_conditional:Nnn \fontspec_if_language:nn {TF,T,F}
106 {
107   \fontspec_if_fontspec_font:TF
108   {
109     \@@_set_font_type:N \font
110     \bool_if:NTF \l_@@_ot_bool
111     {
112       \tl_set:Nn \l_fontspec_script_tl {#1}
113       \@@_iv_str_to_num:Nn \l_@@_script_int {#1}
114       \@@_check_lang:NnTF \font {#2} \prg_return_true: \prg_return_false:
115     }
116     { \prg_return_false: }
117   }
118   { \prg_return_false: }
119 }

```

*(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.

```

120 \prg_new_conditional:Nnn \fontspec_if_current_script:n {TF,T,F}
121 {
122   \fontspec_if_fontspec_font:TF
123   {
124     \@@_set_font_type:N \font
125     \bool_if:NTF \l_@@_ot_bool
126     {
127       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag} \l_@@_tmp_tl
128       \str_if_eq:nVTF {#1} \l_@@_tmp_tl
129       {\prg_return_true:} {\prg_return_false:}
130     }
131     { \prg_return_false: }
132   }
133   { \prg_return_false: }
134 }

```

*(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.

```

135 \prg_new_conditional:Nnn \fontspec_if_current_language:n {TF,T,F}
136 {
137   \fontspec_if_fontspec_font:TF
138   {
139     \@@_set_font_type:N \font
140     \bool_if:NTF \l_@@_ot_bool
141     {
142       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag} \l_@@_tmp_tl
143       \str_if_eq:nVTF {#1} \l_@@_tmp_tl
144       {\prg_return_true:} {\prg_return_false:}
145     }
146   }
147 }

```

```

146     { \prg_return_false: }
147   }
148   { \prg_return_false: }
149 }

```

(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<sup>A</sup>T<sub>E</sub>X family name is automatically generated by fontspec and it's easier to keep it that way.

```

150 \cs_new:Nn \fontspec_set_family:Nnn
151   {
152     \tl_set:Nn \l_@@_family_label_tl {#1}
153     \@@_select_font_family:nn {#2} {#3}
154     \tl_set_eq:NN #1 \l_fontspec_family_tl
155   }
156 \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:NNnn` TODO: the round-about approach of using `\fontname` means that settings such as fontdimens will be lost. (Discovered in unicode-math.) Investigate!

```

157 \cs_new:Nn \fontspec_set_fontface:NNnn
158   {
159     \tl_set:Nn \l_@@_family_label_tl {#1}
160     \@@_select_font_family:nn {#3}{#4}
161     \global \font #1 = \fontname \l_fontspec_font \scan_stop:
162     \tl_set_eq:NN #2 \l_fontspec_family_tl
163   }

```

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

`\fontspec_font_if_exist:n`

```

164 \prg_new_conditional:Nnn \fontspec_font_if_exist:n {TF,T,F}
165   {
166     \group_begin:
167       \@@_init:
168       \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
169       \@@_primitive_font_if_exist:nTF { \@@_construct_font_call:nn {#1} {} }
170       { \group_end: \prg_return_true: }
171       { \group_end: \prg_return_false: }
172     }
173 \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.

```
174 \prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
175 {
176   \exp_args:Nxx \tl_if_in:nnTF
177   { \fontname\font } { \tl_to_str:n {#1} }
178   { \prg_return_true: } { \prg_return_false: }
179 }
```

*(End definition for \fontspec\_if\_current\_feature:nTF. This function is documented on page ??.)*

\fontspec\_if\_small\_caps:TF

```
180 \prg_new_conditional:Nnn \fontspec_if_small_caps: {TF,T,F}
181 {
182   \@@_if_merge_shape:nTF {sc}
183   {
184     \tl_set_eq:Nc \l_@@_smcp_shape_tl { \@@_shape_merge:nn {\f@shape} {sc} }
185   }
186   {
187     \tl_set:Nn \l_@@_smcp_shape_tl {sc}
188   }
189
190   \cs_if_exist:cTF { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
191   {
192     \tl_if_eq:ccTF
193     { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
194     { \f@encoding/\f@family/\f@series/\updefault }
195     { \prg_return_false: }
196     { \prg_return_true: }
197   }
198   { \prg_return_false: }
199 }
```

*(End definition for \fontspec\_if\_small\_caps:TF. This function is documented on page ??.)*

## File X

## fontspec-internal.dtx

## 1 Internals

### 1.1 The main function for setting fonts

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

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

Some often-used variables to know about:

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

```

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

```

```

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

```

(End 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

```

(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_x:nnT { \str_lower_case: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:Nn** Load a possible .fontspec font configuration file. This file could set font-specific options for the font about to be loaded.

```

79 \cs_new:Nn \@@_load_external_fontoptions:Nn
80 {
81   \bool_if:NT \l_@@_fontcfg_bool
82   {
83     <debug> \typeout{: : @@_load_external_fontoptions:Nn \exp_not:N #1 {#2} }
84     \@@_sanitise_fontname:Nn #1 {#2}
85     \tl_set:Nx \l_@@_ext_filename_tl {#1.fontspec}
86     \tl_remove_all:Nn \l_@@_ext_filename_tl {-}
87     \prop_if_in:NVF \g_@@_fontopts_prop #1
88     {
89       \exp_args:No \file_if_exist:nT { \l_@@_ext_filename_tl }
90       { \file_input:n { \l_@@_ext_filename_tl } }
91     }
92   }
93 }

```

*(End definition for \@@\_load\_external\_fontoptions:Nn. This function is documented on page ??.)*

**\@@\_extract\_all\_features:**

```

94 \cs_new:Nn \@@_extract_all_features:n
95 {
96   <debug> \typeout{: : @@_extract_all_features:n { \unexpanded {#1} } }
97   \bool_if:NTF \l_@@_disable_defaults_bool
98   {
99     \clist_set:Nx \l_@@_all_features_clist {#1}
100   }

```

```

101 {
102   \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
103   { \clist_clear:N \l_@@_fontopts_clist }
104
105   \prop_get:NVNF \g_@@_fontopts_prop \l_@@_family_label_tl \l_@@_family_fontopts_clist
106   { \clist_clear:N \l_@@_family_fontopts_clist }
107   \tl_clear:N \l_@@_family_label_tl
108
109   \clist_set:Nx \l_@@_all_features_clist
110   {
111     \g_@@_default_fontopts_clist,
112     \l_@@_family_fontopts_clist,
113     \l_@@_fontopts_clist,
114     #1
115   }
116 }
117 }

```

(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.

```

118 \cs_new:Nn \@@_preparse_features:
119 {
120   <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.

```

121
122   \@@_keys_set_known:nxN {fontspec-preparse-external}
123   { \l_@@_all_features_clist }
124   \l_@@_keys_leftover_clist
125

```

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.

```

126   \tl_set_eq:NN \l_fontspec_fontname_tl \l_@@_fontname_up_tl
127   \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_keys_leftover_clist}
128   \l_@@_keys_leftover_clist
129   \@@_keys_set_known:nxN {fontspec-preparse} {\l_@@_keys_leftover_clist}
130   \l_@@_fontfeat_clist
131 }

```

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

\@@\_load\_font:

```

132 \cs_new:Nn \@@_load_font:
133 {

```

```

134 <debug>\typeout{:: @@_load_font}
135 <debug>\typeout{Set~ base~ font~ for~ preliminary~ analysis: \@@_construct_font_call:nn { \l_
136 \@@_primitive_font_set:Nnn \l_fontspec_font
137 { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} } {\f@size pt}
138 \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-found} {\l_@@_fontn
139 \@@_set_font_type:N \l_fontspec_font
140 <debug>\typeout{Set~ base~ font~ properly: \@@_construct_font_call:nn { \l_@@_fontname_up_tl }
141 \@@_primitive_font_gset:Nnn \l_fontspec_font
142 { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} } {\f@size pt}
143 \l_fontspec_font % this is necessary for LuaLaTeX to check the scripts properly
144 }

```

(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 *<Font features>* are empty and if so don't add in the separator colon.

```

145 \cs_new:Nn \@@_construct_font_call:nnnnnn
146 {
147 <XE> " \@@_fontname_wrap:n { #1 #2 #3 }
148 <LU> " \@@_fontname_wrap:n { #1 #2 } #3
149 #4 #5
150 \str_if_eq:x:nnF {#6}{ } {:#6} "
151 }

```

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

```

152 \cs_new:Nn \@@_construct_font_call:nn
153 {
154 \@@_construct_font_call:nnnnnn
155 {#1}
156 \l_@@_extension_tl
157 \l_@@_ttc_index_tl
158 \l_fontspec_renderer_tl
159 \l_@@_optical_size_tl
160 {#2}
161 }

```

(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'. Xe<sub>La</sub>TeX's syntax is followed since luaotfload provides compatibility.

```

162 \cs_new:Nn \@@_font_is_name:
163 {
164 \cs_set_eq:NN \@@_fontname_wrap:n \use:n
165 }
166 \cs_new:Nn \@@_font_is_file:

```

```

167 {
168   \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
169 }

```

(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.

```

170 \cs_new:Nn \@@_set_scriptlang:
171 {
172   \bool_if:NT \l_@@_firsttime_bool
173   {
174     \tl_if_empty:NTF \l_@@_script_name_tl
175     {
176       \@@_check_script:NnTF \l_fontspec_font {latn}
177       {
178         \tl_set:Nn \l_@@_script_name_tl {Latin}
179         \tl_if_empty:NT \l_@@_lang_name_tl
180         {
181           \tl_set:Nn \l_@@_lang_name_tl {Default}
182         }
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         \@@_info:n {no-scripts}
188       }
189     }
190     {
191       \tl_if_empty:NT \l_@@_lang_name_tl
192       {
193         \tl_set:Nn \l_@@_lang_name_tl {Default}
194       }
195       \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
196       \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
197     }
198   }
199 }

```

(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.

```

200 \cs_new:Nn \@@_get_features:n
201 {
202   <debug> \typeout{: \@@_get_features:Nn { \exp_not:n {#1} } }
203   \@@_init_fontface:
204   \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#1}
205   \l_@@_keys_leftover_clist
206   \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist

```

```

207 <*XE>
208   \bool_if:NTF \l_@@_ot_bool
209   {
210 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
211   % \tracingall
212   \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
213   % \EROROR
214   }
215   {
216 <debug> \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features:~"\l_@@_keys_leftover_c
217   \bool_if:NTF { \l_@@_atsui_bool || \l_@@_graphite_bool }
218   { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
219   }
220 </XE>
221 <*LU>
222 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
223   \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
224 </LU>
225
226   \tl_if_empty:NF \l_@@_mapping_tl
227   { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
228
229   \str_if_eq_x:nnF { \l_@@_hexcol_tl \l_@@_opacity_tl }
230   { \c_@@_hexcol_tl \c_@@_opacity_tl }
231   { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
232   }

```

(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.

```

233 \prg_new_conditional:Nnn \@@_save_family_needed:n { TF }
234 {
235
236 <debug> \typeout{save~ family:~ #1}
237 <debug> \typeout{== fontid_tl: "\l_@@_fontid_tl".}
238
239   \tl_if_empty:NTF \l_@@_nfss_fam_tl
240   {
241     \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
242     {
243       \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
244       \prg_return_false:
245     }
246     {
247       \tl_set:Nx \l_@@_tmp_tl {#1}
248       \tl_remove_all:Nn \l_@@_tmp_tl { ~ }
249       \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl

```

```

250         \prg_return_true:
251     }
252 }
253 {
254     \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
255     \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl _prop }
256     \prg_return_true:
257 }
258 }
259 \cs_new:Nn \@@_save_fontid_family:nn
260 {
261     \prop_get:NnNTF \g_@@_family_int_prop {#2} \l_@@_tmp_tl
262     {
263         \tl_set:Nx \l_@@_tmp_tl
264             { \int_eval:n { \l_@@_tmp_tl + 1 } }
265     }
266     { \tl_set:Nn \l_@@_tmp_tl { 0 } }
267     \prop_gput:NnV \g_@@_family_int_prop {#2} \l_@@_tmp_tl
268     \tl_gset:Nx \g_@@_nfss_family_tl { #2 ( \l_@@_tmp_tl ) }
269     \prop_gput:NnV \g_@@_fontid_family_prop {#1} \g_@@_nfss_family_tl
270 }
271 \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.

```

272 \cs_new:Nn \@@_save_family:nn
273 {
274     \@@_save_fontinfo:n {#2}
275     \@@_find_autofonts:
276     \DeclareFontFamily{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{ }
277     \@@_set_faces:
278     \@@_info:nxx {defining-font} {#1} {#2}
279 }

```

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

**\@@\_save\_fontinfo:n** Saves the relevant font information for future processing.

```

280 \cs_new:Nn \@@_save_fontinfo:n
281 {
282     \prop_new:c {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop}
283     \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
284     \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options} { \l_@@_all_features_ }
285     \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
286     {
287         \@@_construct_font_call:nn {\l_fontspec_fontname_tl}
288         { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
289     }
290     \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
291     \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num} \l_@@_language_int
292     \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_fontspec_script_
293     \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_fontspec_lang_tl
294 }

```

(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 `\bfdefault` is redefined to `b`, all bold shapes defined by this package will also be assigned to `b`.

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

`\@@_find_autofonts:`

```

295 \cs_new:Nn \@@_find_autofonts:
296 {
297   \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
298   {
299     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
300     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
301     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspeg_fontname_tl} {/BI}
302   }
303
304   \bool_if:NF \l_@@_nobf_bool
305   {
306     \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspeg_fontname_tl} {/B}
307   }
308
309   \bool_if:NF \l_@@_noit_bool
310   {
311     \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspeg_fontname_tl} {/I}
312   }
313
314   \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
315 }

```

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

`\@@_set_faces:`

```

316 \cs_new:Nn \@@_set_faces:
317 {
318   \@@_add_nfssfont:nnnn \mddefault \updefault \l_fontspeg_fontname_tl \l_@@_fontfeat_up_
319   \@@_add_nfssfont:nnnn \bfdefault \updefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_clist
320   \@@_add_nfssfont:nnnn \mddefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_clist
321   \@@_add_nfssfont:nnnn \mddefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_clist
322   \@@_add_nfssfont:nnnn \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit_clis
323   \@@_add_nfssfont:nnnn \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl_clis
324
325   \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nnnnn ##2 }
326 }
327 \cs_new:Nn \@@_set_faces_aux:nnnnn
328 {
329   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}

```

```

330 \@@_make_font_shapes:Nnnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
331 }

```

(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").

```

332 \cs_new:Nn \fontspec_complete_fontname:Nn
333 {
334   \tl_set:Nx #1 {#2}
335   \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
336   \LU \tl_remove_all:Nn #1 {-}
337 }

```

(End definition for \fontspec\_complete\_fontname:Nn. This function is documented on page ??.)

\@@\_add\_nfssfont:nnnn #1 : series  
 #2 : shape  
 #3 : fontname  
 #4 : fontspec features

```

338 \cs_new:Nn \@@_add_nfssfont:nnnn
339 {
340   \tl_set:Nx \l_@@_this_font_tl {#3}
341
342   \tl_if_empty:xF {#4}
343   { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
344   { \@@_keys_set_known:nxN {fontspec-preparse-nested} {#4} \l_@@_tmp_tl }
345
346   \tl_if_empty:NF \l_@@_this_font_tl
347   {
348     \prop_put:Nxx \l_@@_nfssfont_prop {#1/#2}
349     { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
350   }
351 }

```

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

### 1.2.1 Fonts

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

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

```

352 \cs_new:Nn \@@_set_font_type:N
353 {
354   \debug \typeout{: : @@@_set_font_type:}
355   \*XE
356   \bool_set_false:N \l_@@_tfm_bool
357   \bool_set_false:N \l_@@_atsui_bool

```

```

358 \bool_set_false:N \l_@@_ot_bool
359 \bool_set_false:N \l_@@_mm_bool
360 \bool_set_false:N \l_@@_graphite_bool
361 \ifcase\XeTeXfonttype #1
362   \bool_set_true:N \l_@@_tfm_bool
363 \or
364   \bool_set_true:N \l_@@_atsui_bool
365   \tl_if_empty:NT \l_fontspec_rendererer_tl { \tl_set:Nn \l_fontspec_rendererer_tl {/AAT} }
366   \ifnum\XeTeXcountvariations #1 > \c_zero
367     \bool_set_true:N \l_@@_mm_bool
368   \fi
369 \or
370   \bool_set_true:N \l_@@_ot_bool
371   \tl_if_empty:NT \l_fontspec_rendererer_tl { \tl_set:Nn \l_fontspec_rendererer_tl {/OT} }
372 \or
373   \bool_set_true:N \l_@@_graphite_bool
374   \tl_if_empty:NT \l_fontspec_rendererer_tl { \tl_set:Nn \l_fontspec_rendererer_tl {/GR} }
375 \fi
376 </XE>

```

If automatic, the `\l_fontspec_rendererer_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:

```

377 <*LU>
378   \bool_set_true:N \l_@@_ot_bool
379 </LU>
380 }

```

(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  $\langle name \rangle$  and  $\langle modifier \rangle$  #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<sub>Y</sub>TeX anyway; todo: test with LuaTeX). If  $\langle font\ name\ tl \rangle$  is not empty, then it's already been specified by the user so abort. If  $\langle Base\ font\ name \rangle$  is not given, we also abort for obvious reasons.

If  $\langle font\ name\ tl \rangle$  is empty, then proceed. If not found,  $\langle font\ name\ tl \rangle$  remains empty. Otherwise, we have a match.

```

381 \cs_new:Nn \@@_set_autofont:Nnn
382 {
383   \bool_if:NF \l_@@_external_bool
384   {
385     \tl_if_empty:xF {#2}
386     {
387       \tl_if_empty:NT #1
388       {
389         \@@_if_autofont:nnTF {#2} {#3}

```

```

390     { \tl_set:Nx #1 {#2#3} }
391     { \@@_info:nx {no-font-shape} {#2#3} }
392   }
393 }
394 }
395 }
396
397 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
398 {
399   \@@_primitive_font_set:Nnn \l_tmpa_font { \@@_construct_font_call:nn {#1} {} } {\f@size pt
400   \@@_primitive_font_set:Nnn \l_tmpb_font { \@@_construct_font_call:nn {#1#2} {} } {\f@size p
401   \str_if_eq_x:nnTF { \fontname \l_tmpa_font } { \fontname \l_tmpb_font }
402   { \prg_return_false: }
403   { \prg_return_true: }
404 }

```

(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.

```

405 \cs_new:Nn \@@_make_font_shapes:Nnnnn
406 {
407   \group_begin:
408     \@@_keys_set_known:nxN {fontspec-preparse-external} { #4 } \l_@@_leftover_clist
409     \@@_load_fontname:n {#1}
410     \@@_declare_shape:nxxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
411   \group_end:
412 }
413
414 \cs_new:Nn \@@_load_fontname:n
415 {
416   <debug> \typeout{:: @@_load_fontname:n {#1} }
417   \@@_load_external_fontoptions:Nn \l_fontspec_fontname_tl {#1}
418   \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
419   { \clist_clear:N \l_@@_fontopts_clist }
420   \keys_set_groups:nnV {fontspec/fontname} {getfontname} \l_@@_fontopts_clist
421   \@@_primitive_font_set:Nnn \l_fontspec_font { \@@_construct_font_call:nn {\l_fontspec_fon
422   \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-found} {#1} }
423 }
424 \keys_define:nn {fontspec/fontname}
425 {
426   Font .tl_set:N = \l_fontspec_fontname_tl ,
427   Font .groups:n = {getfontname} ,
428 }

```

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

`\@@_declare_shape:nnnn` #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>}}`.

```

429 \cs_new:Nn \@@_declare_shape:nnnn
430 {
431   <debug>\typeout{=~ declare_shape:~{\l_fontspeg_fontname_tl}~{#1}~{#2}}
432   \tl_clear:N \l_@@_nfss_tl
433   \tl_clear:N \l_@@_nfss_sc_tl
434   \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspeg_fontname_tl
435
436   \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
437
438   \@@_declare_shapes_normal:nn {#1} {#2}
439   \@@_declare_shapes_smcaps:nn {#1} {#2}
440   \@@_declare_shape_slanted:nn {#1} {#2}
441   \@@_declare_shape_loginfo:nn {#1} {#2}
442 }
443 \cs_generate_variant:Nn \@@_declare_shape:nnnn {nnxx}
  
```

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

`\@@_setup_single_size:nn`

```

444 \cs_new:Nn \@@_setup_single_size:nn
445 {
446   \tl_clear:N \l_@@_size_tl
447   \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
448
449   \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
450   \l_@@_sizing_leftover_clist
451   \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
452   <debug>\typeout{=== size:~\l_@@_size_tl}
453
454   % "normal"
455   \@@_load_fontname:n {\l_@@_sizedfont_tl}
456   \@@_setup_nfss:Nnnn \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
457   <debug> \typeout{=== sized~ font:~ \l_@@_sizedfont_tl}
458
459   % small caps
460   \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist
461
462   \bool_if:NF \l_@@_nosc_bool
463   {
464     \tl_if_empty:NTF \l_@@_fontname_sc_tl
465     {
466       \@@_make_smallcaps:TF
  
```

```

467     {
468 <debug>\typeout{====~Small~ caps~ found.}
469     \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
470     }
471     {
472 <debug>\typeout{====~Small~ caps~ not~ found.}
473     \bool_set_true:N \l_@@_nosc_bool
474     }
475     }
476     { \@@_load_fontname:n {\l_@@_fontname_sc_tl} }% local for each size
477 }
478
479 \bool_if:NF \l_@@_nosc_bool
480 {
481     \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
482     {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
483 }
484 }

```

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

\@@\_setup\_nfss:Nnnn

```

485 \cs_new:Nn \@@_setup_nfss:Nnnn
486 {
487 <debug>\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}
488
489     \@@_get_features:n { #2 , #3 , #4 }
490 <debug>\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist}
491
492     \tl_put_right:Nx #1
493     {
494         <\l_@@_size_tl> \l_@@_scale_tl
495         \@@_construct_font_call:nn { \l_fontspec_fontname_tl }
496         { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
497     }
498 }

```

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

\@@\_declare\_shapes\_normal:nn

```

499 \cs_new:Nn \@@_declare_shapes_normal:nn
500 {
501     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
502     {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
503 }

```

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

\@@\_declare\_shapes\_smcaps:nn

```

504 \cs_new:Nn \@@_declare_shapes_smcaps:nn
505 {
506     \tl_if_empty:NF \l_@@_nfss_sc_tl

```

```

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

```

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

\@@\_DeclareFontShape:nnnnnn

```

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

```

529 \cs_new:Nn \@@_declare_shape_slanted:nn
530 {
531   \bool_if:nT
532   {
533     \str_if_eq_x_p:nn {#2} {\itdefault} &&
534     !(\str_if_eq_x_p:nn {\itdefault} {\sldefault})
535   }
536   {
537     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}
538     {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
539   }
540 }

```

Lastly some informative messaging.

\@@\_declare\_shape\_loginfo:nn

```

541 \cs_new:Nn \@@_declare_shape_loginfo:nn
542 {
543   \tl_gput_right:Nx \g_@@_defined_shapes_tl
544   {

```

```

545 \exp_not:n { \ }
546 -- \exp_not:N \str_case:nn {#1/#2}
547 {
548   {\mddefault/\updefault} {'normal'~}
549   {\bfdefault/\updefault} {'bold'~}
550   {\mddefault/\itdefault} {'italic'~}
551   {\mddefault/\sldefault} {'slanted'~}
552   {\bfdefault/\itdefault} {'bold~ italic'~}
553   {\bfdefault/\sldefault} {'bold~ slanted'~}
554 } (#1/#2)~
555 with~ NFSS~ spec.:~
556 \l_@@_nfss_tl
557 \exp_not:n { \ }
558 -- \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
559 {
560   {\mddefault/\scdefault} {'small~ caps'~}
561   {\bfdefault/\scdefault} {'bold~ small~ caps'~}
562   {\mddefault/\itscdefault} {'italic~ small~ caps'~}
563   {\bfdefault/\itscdefault} {'bold~ italic~ small~ caps'~}
564   {\mddefault/\slscdefault} {'slanted~ small~ caps'~}
565   {\bfdefault/\slscdefault} {'bold~ slanted~ small~ caps'~}
566 }~( #1 / \@@_combo_sc_shape:n {#2} )~
567 with~ NFSS~ spec.:~
568 \l_@@_nfss_sc_tl
569 \tl_if_empty:xF { \l_@@_postadjust_tl }
570 {
571   \exp_not:N \ and~ font~ adjustment~ code: \exp_not:N \ \l_@@_postadjust_tl
572 }
573 }
574 }

```

Maybe \str\_if\_eq\_x:nnF would be better?

## 1.2.2 Features

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

```

\l_@@_pre_feat_sclist
575 \tl_set:Nn \l_@@_pre_feat_sclist
576 {*XE}
577 {
578   \bool_if:NT \l_@@_ot_bool
579   {
580     \tl_if_empty:NF \l_fontspec_script_tl
581     {
582       script = \l_fontspec_script_tl ;
583       language = \l_fontspec_lang_tl ;
584     }
585   }
586 }
587 {/XE}
588 {*LU}
589 {
590   mode = \l_fontspec_mode_tl ;

```

```

591 \tl_if_empty:NF \l_fontspec_script_tl
592 {
593     script = \l_fontspec_script_tl ;
594     language = \l_fontspec_lang_tl ;
595 }
596 }
597 </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF
598 <LU>\cs_new:Nn \@@_make_smallcaps:TF
599 <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
600 {
601     \@@_check_ot_feat:NnTF \l_fontspec_font {smcp} {#1} {#2}
602 }
603 <*XE>
604 \cs_new:Nn \@@_make_smallcaps:TF
605 {
606     \bool_if:NTF \l_@@_ot_bool
607     { \@@_make_ot_smallcaps:TF {#1} {#2} }
608     {
609         \bool_if:NT \l_@@_atsui_bool
610         { \@@_make_AAT_feature_string:NnnTF \l_fontspec_font {3}{3} {#1} {#2} }
611     }
612 }
613 </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

```

614 \cs_new:Nn \@@_update_featstr:n
615 {
616     <debug> \typeout{::: @@_update_featstr:n {#1}}
617     \bool_if:NF \l_@@_firsttime_bool
618     {
619         \tl_gset:Nx \g_@@_single_feat_tl { #1 }
620     <debug> \typeout{:::~ Adding~ feature.}
621     \tl_gput_right:Nx \g_@@_rawfeatures_sclist {#1;}
622     }
623 }

```

\@@\_remove\_clashing\_featstr:n

```

624 \cs_new:Nn \@@_remove_clashing_featstr:n
625 {
626     <debug> \typeout{::: @@_remove_clashing_featstr:n {#1}}
627     \clist_map_inline:nn {#1}
628     {
629     <debug> \typeout{:::~ Removing~ feature~ "##1;" }
630     \tl_gremove_all:Nn \g_@@_rawfeatures_sclist {##1;}
631     }
632 }

```

### 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:
633 \cs_set:Npn \@@_init:
634 {
635   \debug \typeout{: \@@_init:}
636   \bool_set_false:N \l_@@_ot_bool
637   \bool_set_true:N \l_@@_firsttime_bool
638   \@@_font_is_name:
639   \tl_clear:N \l_@@_font_path_tl
640   \tl_clear:N \l_@@_optical_size_tl
641   \tl_clear:N \l_@@_ttc_index_tl
642   \tl_clear:N \l_fontspeg_renderer_tl
643   \tl_gclear:N \g_@@_defined_shapes_tl
644   \tl_gclear:N \g_@@_curr_series_tl
645   \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspeg_encoding_tl
646   \*LU
647   \tl_set:Nn \l_fontspeg_mode_tl {node}
648   \int_set:Nn \luatex_prehyphenchar:D { \- } % fixme
649   \int_zero:N \luatex_posthyphenchar:D % fixme
650   \int_zero:N \luatex_preexhyphenchar:D % fixme
651   \int_zero:N \luatex_postexhyphenchar:D % fixme
652   \*LU
653 }
```

Executed in \@@\_get\_features:Nn.

```
\@@_init_fontface:
654 \cs_new:Nn \@@_init_fontface:
655 {
656   \tl_gclear:N \g_@@_rawfeatures_sclist
657   \tl_clear:N \l_@@_scale_tl
658   \tl_set_eq:NN \l_@@_opacity_tl \c_@@_opacity_tl
659   \tl_set_eq:NN \l_@@_hexcol_tl \c_@@_hexcol_tl
660   \tl_set_eq:NN \l_@@_postadjust_tl \c_@@_postadjust_tl
661   \tl_clear:N \l_@@_wordspace_adjust_tl
662   \tl_clear:N \l_@@_punctspace_adjust_tl
663 }
```

### 1.4 Miscellaneous

This macro takes a four character string and converts it to the numerical representation required for X<sub>Y</sub>TeX OpenType script/language/feature purposes. The output is stored in #1.

The reason it's ugly is because 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, delimited by a space; this input is 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 -.

```
664 \cs_new:Nn \@@_iv_str_to_num:Nn
665 {
```

```

666     \@@_strip_leading_sign:Nw #1#2 \q_nil
667   }
668 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
669   {
670     \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
671       { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
672       { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
673   }
674 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
675   {
676     \int_set:Nn #1
677     {
678       `#2 * "10000000
679       + `#3 * "10000
680       + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
681       + \ifx \c_empty_tl #5 32 \else `#5 \fi
682     }
683   }
684 \cs_generate_variant:Nn \@@_iv_str_to_num:Nn {No}

```

## File XI

# fontspec-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: }
4 }

#1 : Feature key
\@@_define_opentype_feature:nnnnn #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:nnnnn
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     }
25   }
26   {
27     \keys_define:nn {fontspec-opentype}
28     {
29       #1/#2 .code:n =
30       {
31         <debug> \typeout{::::::::::fontspec-opentype~#1/#2~::~#3/#4/#5}
32         \@@_make_OT_feature:nnn {#3} {#4} {#5}
33       }
34     }
35   }
36 }
```

```

#1 : Feature key
\@@_define_opentype_onoffreset:nnnnm #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)

37 \cs_new:Nn \@@_feat_off:n {#10ff}
38 \cs_new:Nn \@@_feat_reset:n {#1Reset}

39 \cs_new:Nn \@@_define_opentype_onoffreset:nnnnn
40 {
41   \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} {#2} {#3} {+#4} {#5}
42   \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_off:n {#2} } {#3} {-#4} {
43   \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {+#4,-
44 }

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

45 \cs_new:Nn \@@_define_opentype_onreset:nnnnn
46 {
47   \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} {#2} {#3} {#4} {#5}
48   \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {#4}
49 }

```

## 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.

```

50 \cs_new:Nn \@@_make_OT_feature:nnn
51 {
52   <debug> \typeout{: @@@_make_OT_feature:nnn \exp_not:n { {#1}{#2}{#3} } }
53
54   \bool_set_true:N \l_@@_proceed_bool
55   \bool_set_true:N \l_@@_check_feat_bool
56
57   \tl_if_empty:nT {#1} { \bool_set_false:N \l_@@_check_feat_bool }
58   \bool_if:NT \l_@@_check_feat_bool
59   {
60     \@@_check_ot_feat:NnF \l_fontspeg_font {#1}
61     {
62       \@@_warning:nx {icu-feature-not-exist-in-font} {#1}
63       \bool_set_false:N \l_@@_proceed_bool
64     }
65   }

```

```

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

```

(End definition for \@@\_DeclareFontShape:nnnnnn 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. The output boolean is \tempwatrue. \l\_@@\_script\_int is used to store the number corresponding to the script tag string.

```

79 \prg_new_conditional:Nnn \@@_check_script:Nn {TF}
80 {
81 \bool_if:NTF \l_@@_never_check_bool
82 { \prg_return_true: }
83 <*XE>
84 {
85 \@@_iv_str_to_num:Nn \l_@@_strnum_int {#2}
86 \int_set:Nn \l_tmpb_int { \XeTeXOTcountscripts #1 }
87 \int_zero:N \l_tmpa_int
88 \bool_set_false:N \l__fontspec_check_bool
89 \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
90 {
91 \ifnum \XeTeXOTscripttag #1 \l_tmpa_int = \l_@@_strnum_int
92 \bool_set_true:N \l__fontspec_check_bool
93 \int_set:Nn \l_tmpa_int { \l_tmpb_int }
94 }
95 \else
96 \int_incr:N \l_tmpa_int
97 \fi
98 \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
99 }
100 </XE>
101 <*LU>
102 {
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 \directlua{fontspec.check_ot_script("\l_@@_tmp_tl", "#2")}
107 \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
108 }
109 </LU>
110 }

```

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

**\@@\_check\_lang:NnTF** This macro takes an OpenType language tag and checks if it exists in the current font/script. The output boolean is \@tempwattrue. \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'.

```

111 \prg_new_conditional:Nnn \@@_check_lang:Nn {TF}
112 {
113   \bool_if:NTF \l_@@_never_check_bool
114   { \prg_return_true: }
115 <*XE>
116 {
117   \@@_iv_str_to_num:Nn \l_@@_strnum_int {#2}
118   \int_set:Nn \l_tmpb_int
119   { \XeTeXOTcountlanguages #1 \l_@@_script_int }
120   \int_zero:N \l_tmpa_int
121   \bool_set_false:N \l__fontspec_check_bool
122   \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
123   {
124     \ifnum\XeTeXOTlanguage tag #1 \l_@@_script_int \l_tmpa_int = \l_@@_strnum_int
125     \bool_set_true:N \l__fontspec_check_bool
126     \int_set:Nn \l_tmpa_int {\l_tmpb_int}
127   }
128   \else
129     \int_incr:N \l_tmpa_int
130   \fi
131   \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
132 }
133 </XE>
134 <*LU>
135 {
136   \cs_if_eq:NNTF #1 \font
137   { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
138   { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
139   \directlua
140   {
141     fontspec.check_ot_lang( "\l_@@_tmp_tl", "#2", "\l_fontspec_script_tl" )
142   }
143   \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
144 }
145 </LU>
146 }

```

(End definition for \@@\_check\_lang:NnTF. This function is 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. \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'.

```

147 \prg_new_conditional:Nnn \@@_check_ot_feat:Nn {TF,F}
148 {

```

```

149     \bool_if:NTF \l_@@_never_check_bool
150     { \prg_return_true: }
151 <*XE>
152 {
153 <debug>\typeout{::~ fontspec_check_ot_feat:n~ {#1}}
154     \int_set:Nn \l_tmpb_int
155     {
156         \XeTeXOTcountfeatures #1
157             \l_@@_script_int
158             \l_@@_language_int
159     }
160     \@@_iv_str_to_num:Nn \l_@@_strnum_int {#2}
161     \int_zero:N \l_tmpa_int
162     \bool_set_false:N \l_@@_check_bool
163     \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
164     {
165         \ifnum\XeTeXOTfeaturetag #1 \l_@@_script_int \l_@@_language_int
166             \l_tmpa_int = \l_@@_strnum_int
167             \bool_set_true:N \l_@@_check_bool
168             \int_set:Nn \l_tmpa_int { \l_tmpb_int }
169         \else
170             \int_incr:N \l_tmpa_int
171         \fi
172     }
173     \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
174 }
175 </XE>
176 <*LU>
177 {
178 <debug>\typeout{::~ fontspec_check_ot_feat:n~ {#1}}
179     \cs_if_eq:NNTF #1 \font
180     { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
181     { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
182     \directlua
183     {
184         fontspec.check_ot_feat(
185             "\l_@@_tmp_tl", "#2",
186             "\l_fontspeg_lang_tl", "\l_fontspeg_script_tl"
187         )
188     }
189     \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
190 }
191 </LU>
192 }

```

(End definition for \@@\_check\_ot\_feat:NnTF. This function is documented on page ??.)

## 1.2 OpenType feature information

```

193 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {aalt}{Access-All-Alternates}
194 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvf}{Above-base-Forms}
195 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvm}{Above-base-Mark-Positioning}

```

196 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {abvs}{Above-base-Substitutions}  
 197 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {afrc}{Alternative-Fractions}  
 198 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {akhn}{Akhands}  
 199 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {blwf}{Below-base-Forms}  
 200 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {blwm}{Below-base-Mark-Positioning}  
 201 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {blws}{Below-base-Substitutions}  
 202 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {calt}{Contextual~Alternates}  
 203 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {case}{Case-Sensitive~Forms}  
 204 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ccmp}{Glyph-Composition~/~Decomposition}  
 205 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {cfar}{Conjunct~Form~After~Ro}  
 206 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {cjct}{Conjunct~Forms}  
 207 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {clig}{Contextual~Ligatures}  
 208 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {cpct}{Centered-CJK-Punctuation}  
 209 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {cpsp}{Capital-Spacing}  
 210 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {cswl}{Contextual-Swash}  
 211 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {curs}{Cursive-Positioning}  
 212 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {cvNN}{Character~Variant~\$N\$}  
 213 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {c2pc}{Petite-Capitals-From-Capitals}  
 214 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {c2sc}{Small~Capitals~From~Capitals}  
 215 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {dist}{Distances}  
 216 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {dlig}{Discretionary~Ligatures}  
 217 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {dnom}{Denominators}  
 218 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {dtls}{Dotless~Forms}  
 219 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {expt}{Expert~Forms}  
 220 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {falt}{Final~Glyph-on-Line~Alternates}  
 221 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {fin2}{Terminal~Forms~\#2}  
 222 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {fin3}{Terminal~Forms~\#3}  
 223 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {fina}{Terminal~Forms}  
 224 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {flac}{Flattened~accent~forms}  
 225 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {frac}{Fractions}  
 226 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {fwid}{Full~Widths}  
 227 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {half}{Half~Forms}  
 228 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {haln}{Halant~Forms}  
 229 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {halt}{Alternate~Half~Widths}  
 230 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {hist}{Historical~Forms}  
 231 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {hkna}{Horizontal-Kana~Alternates}  
 232 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {hlig}{Historical~Ligatures}  
 233 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {hngl}{Hangul}  
 234 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {hojo}{Hojo~Kanji~Forms}  
 235 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {hwid}{Half~Widths}  
 236 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {init}{Initial~Forms}  
 237 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {isol}{Isolated~Forms}  
 238 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ital}{Italics}  
 239 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {jalt}{Justification~Alternates}  
 240 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {jp78}{JIS78~Forms}  
 241 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {jp83}{JIS83~Forms}  
 242 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {jp90}{JIS90~Forms}  
 243 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {jp04}{JIS2004~Forms}  
 244 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {kern}{Kerning}  
 245 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {lfbd}{Left~Bounds}  
 246 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {liga}{Standard~Ligatures}

247 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ljmo}{Leading~Jamo~Forms}  
 248 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {lnum}{Lining~Figures}  
 249 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {loc1}{Localized~Forms}  
 250 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ltra}{Left-to-right~alternates}  
 251 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ltrm}{Left-to-right~mirrored~forms}  
 252 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {mark}{Mark~Positioning}  
 253 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {med2}{Medial~Forms~\#2}  
 254 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {medi}{Medial~Forms}  
 255 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {mgrk}{Mathematical~Greek}  
 256 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {mkmk}{Mark~to~Mark~Positioning}  
 257 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {mset}{Mark~Positioning~via~Substitution}  
 258 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {nalt}{Alternate~Annotation~Forms}  
 259 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {nlck}{NLC~Kanji~Forms}  
 260 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {nukt}{Nukta~Forms}  
 261 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {numr}{Numerators}  
 262 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {onum}{Oldstyle~Figures}  
 263 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {opbd}{Optical~Bounds}  
 264 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ordn}{Ordinals}  
 265 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ornm}{Ornaments}  
 266 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {palt}{Proportional~Alternate~Widths}  
 267 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {pcap}{Petite~Capitals}  
 268 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {pkna}{Proportional~Kana}  
 269 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {pnun}{Proportional~Figures}  
 270 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {pref}{Pre~Base~Forms}  
 271 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {pres}{Pre~base~Substitutions}  
 272 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {pstf}{Post~base~Forms}  
 273 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {psts}{Post~base~Substitutions}  
 274 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {pwid}{Proportional~Widths}  
 275 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {qwid}{Quarter~Widths}  
 276 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rand}{Randomize}  
 277 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rc1t}{Required~Contextual~Alternates}  
 278 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rkrf}{Rakar~Forms}  
 279 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {r1ig}{Required~Ligatures}  
 280 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rphf}{Reph~Forms}  
 281 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rtbd}{Right~Bounds}  
 282 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rtla}{Right-to-left~alternates}  
 283 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rtlm}{Right-to-left~mirrored~forms}  
 284 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ruby}{Ruby~Notation~Forms}  
 285 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {rvrn}{Required~Variation~Alternates}  
 286 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {salt}{Stylistic~Alternates}  
 287 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {sinf}{Scientific~Inferiors}  
 288 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {size}{Optical~size}  
 289 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {smcp}{Small~Capitals}  
 290 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {smpl}{Simplified~Forms}  
 291 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ssNN}{Stylistic~Set~\$N\$}  
 292 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {ssty}{Math~script~style~alternates}  
 293 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {stch}{Stretching~Glyph~Decomposition}  
 294 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {subs}{Subscript}  
 295 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {supr}{Superscript}  
 296 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {swsh}{Swash}  
 297 \prop\_gput:Nnn \g\_@@\_all\_opentype\_feature\_names\_prop {titl}{Titling}

```

298 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing~Jamo~Forms}
299 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional~Name~Forms}
300 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular~Figures}
301 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {trad}{Traditional~Forms}
302 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third~Widths}
303 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {unic}{Unicase}
304 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate~Vertical~Metrics}
305 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu~Variants}
306 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical~Writing}
307 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate~Vertical~Half~Metrics}
308 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vjmo}{Vowel~Jamo~Forms}
309 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical~Kana~Alternates}
310 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkern}{Vertical~Kerning}
311 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional~Alternate~Vertical~Metrics}
312 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrt2}{Vertical~Alternates~and~Rotation}
313 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrtr}{Vertical~Alternates~for~Rotation}
314 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {zero}{Slashed~Zero}

```

## File XII

# fontspec-graphite.dtx

## 1 Graphite/AAT code

`\@@_define_aat_feature_group:n`

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

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

`\@@_define_aat_feature:nnnn`

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

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

`\@@_make_AAT_feature:nn`

```
10 \cs_new:Nn \@@_make_AAT_feature:nn
11 {
12   \tl_if_empty:nTF {#1}
13   { \@@_warning:n {aat-feature-not-exist} }
14   {
15     \@@_make_AAT_feature_string:NnnTF \l_fontspec_font {#1}{#2}
16     {
17       \@@_update_featstr:n {\l_fontspec_feature_string_tl}
18     }
19     { \@@_warning:nx {aat-feature-not-exist-in-font} {#1,#2} }
20   }
21 }
```

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

`\@@_make_AAT_feature_string:NnnTF`

This macro takes the numerical codes for a font feature and creates a specified macro containing the string required in the font definition to turn that feature on or off. Used primarily in [...], but also used to check if small caps exists in the requested font (see page 56).

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<sub>Y</sub>T<sub>E</sub>X 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`.

```
22 \prg_new_conditional:Nnn \@@_make_AAT_feature_string:Nnn {TF,T,F}
```

```

23 {
24   \tl_set:Nx \l_@@_tmpa_tl { \XeTeXfeaturename #1 #2 }
25   \tl_if_empty:NTF \l_@@_tmpa_tl
26   { \prg_return_false: }
27   {
28     \int_compare:nTF { \XeTeXisexclusivefeature #1 #2 > 0 }
29     {
30       \tl_set:Nx \l_@@_tmpb_tl { \XeTeXselectorname #1 #2 \space #3 }
31     }
32     {
33       \int_if_even:nTF {#3}
34       {
35         \tl_set:Nx \l_@@_tmpb_tl { \XeTeXselectorname #1 #2 \space #3 }
36       }
37       {
38         \tl_set:Nx \l_@@_tmpb_tl
39         {
40           \XeTeXselectorname #1 #2 \space \numexpr#3-1\relax
41         }
42         \tl_if_empty:NF \l_@@_tmpb_tl { \tl_put_left:Nn \l_@@_tmpb_tl {!} }
43       }
44     }
45     \tl_if_empty:NTF \l_@@_tmpb_tl
46     { \prg_return_false: }
47     {
48       \tl_set:Nx \l_fontspec_feature_string_tl { \l_@@_tmpa_tl = \l_@@_tmpb_tl }
49       \prg_return_true:
50     }
51   }
52 }

```

(End definition for \@@\_make\_AAT\_feature\_string:NnnTF. This function is documented on page ??.)

## File XIII

# fontspec-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-n
5   fontspec-renderer
6 }
```

Wrapper function to save some characters in the source:

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

For catching features that cannot be used in \addfontfeatures:

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

### 1.1 Pre-pre-parsing stages

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

#### Don't load font config file

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

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

```
24 \@@_keys_define_code:nnn {fontspec-preparse-external} {Path}
25 {
26   \bool_set_true:N \l_@@_nobf_bool
27   \bool_set_true:N \l_@@_noit_bool
28   \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 even whether certain features are available.

```

47 \keys_define:nn {fontspec-renderer}
48 {
49 Renderer .choices:nn =
50 {AAT,ICU,OpenType,Graphite,Full,Basic}
51 {
52 \int_compare:nTF {\l_keys_choice_int <= 4} {
53 <*XE>
54 \tl_set:Nx \l_fontspec_renderer_tl
55 {
56 \int_case:nn \l_keys_choice_int { 1 {/AAT} 2 {/OT} 3 {/OT} 4 {/GR} }
57 }
58 \tl_gset:Nx \g_@@_single_feat_tl { \l_fontspec_renderer_tl }
59 </XE>
60 <*LU>
61 \@@_warning:nx {only-xetex-feature} {Renderer=AAT/OpenType/Graphite}
62 </LU>
63 }
64 {
65 <*XE>
66 \@@_warning:nx {only-luatex-feature} {Renderer=Full/Basic}
67 </XE>
68 <*LU>
69 \tl_set:Nx \l_fontspec_mode_tl
70 {
71 \int_case:nn \l_keys_choice_int { 5 {node} 6 {base} }

```

```

72     }
73     \tl_gset:Nx \g_@@_single_feat_tl { mode=\l_fontspec_mode_tl }
74 </LU>
75   }
76 }
77 }

```

## 1.2 Pre-parsed features

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

```

78 \@@_keys_define_code:nnn {fontspec-prepare} {Script}
79 {
80 <XE> \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
81 \tl_set:Nn \l_@@_script_name_tl {#1}
82 }

```

Exactly the same:

```

83 \@@_keys_define_code:nnn {fontspec-prepare} {Language}
84 {
85 <XE> \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
86 \tl_set:Nn \l_@@_lang_name_tl {#1}
87 }

```

## TTC font index

```

88 \@@_keys_define_code:nnn {fontspec-prepare} {FontIndex}
89 {
90 \str_if_eq:x:nnF { \str_lower_case:f {\l_@@_extension_tl} } {.ttc}
91 { \@@_warning:n {font-index-needs-ttc} }
92 <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
93 <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
94 }
95 \@@_keys_define_code:nnn {fontspec} {FontIndex}
96 {
97 <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
98 <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
99 }

```

## 1.3 Font faces

### Upright

```

100 \@@_keys_define_code:nnn {fontspec-prepare-external} {UprightFont}
101 {
102 \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
103 }

```

## Italic and slanted

```
104 \@@_keys_define_code:nnn {fontspec-preparse-external} {ItalicFont}
105 {
106   \tl_if_empty:nTF {#1}
107   {
108     \bool_set_true:N \l_@@_noit_bool
109   }
110   {
111     \bool_set_false:N \l_@@_noit_bool
112     \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
113   }
114 }
115 \@@_keys_define_code:nnn {fontspec-preparse-external} {SlantedFont}
116 {
117   \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
118 }
```

**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!

```
119 %\@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSeries}
120 % {
121 %   \tl_gset:Nx \g_@@_curr_series_tl { #1 }
122 %   \seq_put_right:Nx \l_@@_bf_series_seq { #1 }
123 % }
```

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

```
124 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldFont}
125 {
126   \tl_if_empty:nTF {#1}
127   {
128     \bool_set_true:N \l_@@_nobf_bool
129   }
130   {
131     \bool_set_false:N \l_@@_nobf_bool
132     \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
133
134     \seq_if_empty:NT \l_@@_bf_series_seq
135     {
136       \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
137       \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
138     }
139
140     \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
141     {
142       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
143     }
144
145     \prop_put:NxV \l_@@_nfss_prop {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
146
147     <debug>\typeout{Setting~bold~font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
```

```

148     }
149 }
150 }

```

### Bold italic/slanted

```

151 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldItalicFont}
152 {
153   \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
154 }
155 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldSlantedFont}
156 {
157   \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
158 }

```

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

```

159 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
160 {
161   \tl_if_empty:nTF {#1}
162   {
163     \bool_set_true:N \l_@@_nosc_bool
164   }
165   {
166     \bool_set_false:N \l_@@_nosc_bool
167     \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
168   }
169 }

```

### 1.3.1 Prepared font features

```

170 \@@_keys_define_code:nnn {fontspec-prepare} {UprightFeatures}
171 {
172   \clist_set:Nn \l_@@_fontfeat_up_clist {#1}
173 }
174 \@@_keys_define_code:nnn {fontspec-prepare} {BoldFeatures}
175 {
176   \clist_set:Nn \l_@@_fontfeat_bf_clist {#1}
177 }
178 % \prop_put:NxV \l_@@_nfss_prop
179 % {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
180 }
181 \@@_keys_define_code:nnn {fontspec-prepare} {ItalicFeatures}
182 {
183   \clist_set:Nn \l_@@_fontfeat_it_clist {#1}
184 }
185 \@@_keys_define_code:nnn {fontspec-prepare} {BoldItalicFeatures}
186 {
187   \clist_set:Nn \l_@@_fontfeat_bfit_clist {#1}
188 }
189 \@@_keys_define_code:nnn {fontspec-prepare} {SlantedFeatures}
190 {

```

```

191 \clist_set:Nn \l_@@_fontfeat_sl_clist {#1}
192 }
193 \@@_keys_define_code:nnn {fontspec-preparse} {BoldSlantedFeatures}
194 {
195 \clist_set:Nn \l_@@_fontfeat_bfsl_clist {#1}
196 }

```

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

```

197 \@@_keys_define_code:nnn {fontspec} {SmallCapsFeatures}
198 {
199 \bool_if:NF \l_@@_firsttime_bool
200 {
201 \clist_set:Nn \l_@@_fontfeat_sc_clist {#1}
202 }
203 }

```

### Features varying by size

```

204 \@@_keys_define_code:nnn {fontspec-preparse} {SizeFeatures}
205 {
206 \clist_set:Nn \l_@@_sizefeat_clist {#1}
207 \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
208 }
209 \@@_keys_define_code:nnn {fontspec-preparse-nested} {SizeFeatures}
210 {
211 \clist_set:Nn \l_@@_sizefeat_clist {#1}
212 \tl_if_empty:NT \l_@@_this_font_tl
213 { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
214 }
215 \@@_keys_define_code:nnn {fontspec-preparse-nested} {Font}
216 {
217 \tl_set:Nn \l_@@_this_font_tl {#1}
218 }
219 \@@_keys_define_code:nnn {fontspec} {SizeFeatures}
220 {
221 % dummy
222 }
223 \@@_keys_define_code:nnn {fontspec} {Font}
224 {
225 % dummy
226 }
227 \@@_keys_define_code:nnn {fontspec-sizing} {Size}
228 {
229 \tl_set:Nn \l_@@_size_tl {#1}
230 }
231 \@@_keys_define_code:nnn {fontspec-sizing} {Font}
232 {
233 \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
234 }

```

## 1.4 General font-independent features

These features can be applied to any font.

**NFSS encoding** For the very brave.

```

235 \@@_keys_define_code:nnn {fontspec-preparse} {NFSSEncoding}
236 {
237   \tl_gset:Nx \g_@@_nfss_enc_tl { #1 }
238 }

```

**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.)

```

239 \@@_keys_define_code:nnn {fontspec-preparse} {NFSSFamily}
240 {
241   \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
242 }

```

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

```

243 \@@_keys_define_code:nnn {fontspec} {FontFace}
244 {
245   \tl_clear:N \l_@@_this_font_tl
246   \clist_set:No \l_@@_arg_clist { \use_iii:nnn #1 }
247   \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
248   \int_compare:N { \clist_count:N \l_@@_arg_clist = 1 }
249   {
250     <debug>\typeout{FontFace~ parsing:~ one~ clist~ item}
251     \tl_if_in:NnF \l_@@_arg_clist {=}
252     {
253       <debug>\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
254       \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
255       \tl_clear:N \l_@@_this_feat_clist
256     }
257   }
258
259   \@@_add_nfssfont:nnnn
260   {\use_i:nnn #1} {\use_ii:nnn #1} {\l_@@_this_font_tl} {\l_@@_this_feat_clist}
261 }

```

**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.

```

262 \@@_keys_define_code:nnn {fontspec} {Scale}
263 {
264   \str_case:nnF {#1}
265   {
266     {MatchLowercase} { \@@_calc_scale:n {5} }
267     {MatchUppercase} { \@@_calc_scale:n {8} }
268   }
269   { \tl_set:Nx \l_@@_scale_tl {#1} }
270   \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
271 }

```

`\@@_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<sub>Y</sub>TeX).

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

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

```

272 \cs_new:Nn \@@_calc_scale:n
273 {
274   \group_begin:
275
276   \fontencoding { \encodingdefault }
277   \fontfamily { \rmdefault }
278   \selectfont
279
280   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
281   \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_fontspeg_font
282
283   \tl_gset:Nx \l_@@_scale_tl
284   {
285     \fp_eval:n { \dim_to_fp:n { \l_@@_tmpa_dim } /
286                 \dim_to_fp:n { \l_@@_tmpb_dim } }
287   }
288
289   \@@_info:n {set-scale}
290   \group_end:
291 }

```

*(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 .t<sub>fm</sub> 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'.

```

292 \cs_new:Nn \@@_set_font_dimen:NnN
293 {
294   \dim_set:Nn #1 { \fontdimen #2 #3 }
295   \dim_compare:nNnT #1 = {0pt}
296   {
297     \settoheight #1
298     {
299       \str_if_eq:nnTF {#3} {\font} \rmfamily #3
300       \int_case:nnF #2
301       {
302         {5} {x} % x-height
303         {8} {X} % cap-height
304       } {?} % "else" clause; never reached.
305     }
306   }
307 }

```

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

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

```

308 \@@_keys_define_code:nnn {fontspec} {WordSpace}
309 {
310   \bool_if:NF \l_@@_firsttime_bool
311   { \_fontspec_parse_wordspace:w #1,,,\q_stop }
312 }
313 \@@_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}.

```

314 \cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
315 {
316   \tl_if_empty:nTF {#4}
317   {
318     \tl_set:Nn \l_@@_wordspace_adjust_tl
319     {
320       \fontdimen 2 \font = #1 \fontdimen 2 \font
321       \fontdimen 3 \font = #1 \fontdimen 3 \font
322       \fontdimen 4 \font = #1 \fontdimen 4 \font
323     }
324   }
325   {
326     \tl_set:Nn \l_@@_wordspace_adjust_tl
327     {
328       \fontdimen 2 \font = #1 \fontdimen 2 \font
329       \fontdimen 3 \font = #2 \fontdimen 3 \font
330       \fontdimen 4 \font = #3 \fontdimen 4 \font
331     }
332   }
333 }

```

(End definition for \\_fontspec\_parse\_wordspace:w. This function is documented on page ??.)

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

```

334 \@@_keys_define_code:nnn {fontspec} {PunctuationSpace}
335 {
336   \str_case_x:nnF {#1}
337   {
338     {WordSpace}
339     {
340       \tl_set:Nn \l_@@_punctspace_adjust_tl
341       { \fontdimen 7 \font = 0 \fontdimen 2 \font }
342     }
343     {TwiceWordSpace}
344     {
345       \tl_set:Nn \l_@@_punctspace_adjust_tl
346       { \fontdimen 7 \font = 1 \fontdimen 2 \font }
347     }
348   }

```

```

349 {
350   \tl_set:Nn \l_@@_punctspace_adjust_tl
351   { \fontdimen 7 \font = #1 \fontdimen 7 \font }
352 }
353 }
354 \@@_aff_error:n {PunctuationSpace}

```

### Secret hook into the font-adjustment code

```

355 \@@_keys_define_code:nnn {fontspec} {FontAdjustment}
356 {
357   \tl_put_right:Nx \l_@@_postadjust_tl {#1}
358 }

```

### Letterspacing

```

359 \@@_keys_define_code:nnn {fontspec} {LetterSpace}
360 {
361   \@@_update_featstr:n {letterspace=#1}
362 }

```

**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.

```

363 \@@_keys_define_code:nnn {fontspec} {HyphenChar}
364 {
365   \str_if_eq:nnTF {#1} {None}
366   {
367     \tl_put_right:Nn \l_@@_postadjust_tl
368     { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
369   }
370   {
371     \@@_warning:nx {only-xetex-feature} {HyphenChar}
372
373     \tl_if_single:nTF {#1}
374     { \tl_set:Nn \l_fontspec_hyphenchar_tl {`#1} }
375     { \tl_set:Nn \l_fontspec_hyphenchar_tl { #1} }
376
377     \@@_primitive_font_glyph_if_exist:NnTF \l_fontspec_font {\l_fontspec_hyphenchar_tl}
378     {
379       \tl_put_right:Nn \l_@@_postadjust_tl
380       { \@@_primitive_font_set_hyphenchar:Nn \font { \l_fontspec_hyphenchar_tl } }
381     }
382     { \@@_error:nx {no-glyph}{#1} }
383
384   }
385 }
386 \@@_aff_error:n {HyphenChar}

```

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

```

387 \@@_keys_define_code:nnn {fontspec} {Color}
388 {
389   \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
390   {
391     \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
392   }
393   {
394     \int_compare:nTF { \tl_count:n {#1} == 6 }
395     { \tl_set:Nn \l_@@_hexcol_tl {#1} }
396     {
397       \int_compare:nTF { \tl_count:n {#1} == 8 }
398       { \fontspec_parse_colour:viii #1 }
399       {
400         \bool_if:NF \l_@@_firsttime_bool
401         { \@@_warning:nx {bad-colour} {#1} }
402       }
403     }
404   }
405 }

406 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
407 {
408   \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
409   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
410   {
411     \bool_if:NF \l_@@_firsttime_bool
412     { \@@_warning:nx {opa-twice-col} {#7#8} }
413   }
414   \tl_set:Nn \l_@@_opacity_tl {#7#8}
415 }
416 \aliasfontfeature{Color}{Colour}

417 \@@_keys_define_code:nnn {fontspec} {Opacity}
418 {
419   \int_set:Nn \l_@@_tmp_int {255}
420   \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
421   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
422   {
423     \bool_if:NF \l_@@_firsttime_bool
424     { \@@_warning:nx {opa-twice} {#1} }
425   }
426   \tl_set:Nx \l_@@_opacity_tl
427   {
428     \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
429     \int_to_hex:n { \l_@@_tmp_int }
430   }
431 }

```

## Mapping

```

432 <*XE>
433 \@@_keys_define_code:nnn {fontspec-aat} {Mapping}

```

```

434 {
435   \tl_set:Nn \l_@@_mapping_tl { #1 }
436 }
437 \@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
438 {
439   \tl_set:Nn \l_@@_mapping_tl { #1 }
440 }
441 \</XE>
442 \< *LU>
443 \@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
444 {
445   \str_if_eq:nnTF {#1} {tex-text}
446   {
447     \@@_warning:n {no-mapping-ligtext}
448     \msg_redirect_name:nnn {fontspec} {no-mapping-ligtext} {none}
449     \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
450   }
451   { \@@_warning:n {no-mapping} }
452 }
453 \</LU>

```

#### 1.4.1 Continuous font axes

```

454 \@@_keys_define_code:nnn {fontspec} {Weight}
455 {
456   \@@_update_featstr:n{weight=#1}
457 }
458 \@@_keys_define_code:nnn {fontspec} {Width}
459 {
460   \@@_update_featstr:n{width=#1}
461 }
462 \@@_keys_define_code:nnn {fontspec} {OpticalSize}
463 \< *XE>
464 {
465   \bool_if:NTF \l_@@_ot_bool
466   {
467     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
468   }
469   {
470     \bool_if:NT \l_@@_mm_bool
471     {
472       \@@_update_featstr:n { optical size = #1 }
473     }
474   }
475   \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
476   {
477     \bool_if:NT \l_@@_firsttime_bool
478     { \@@_warning:n {no-opticals} }
479   }
480 }
481 \</XE>
482 \< *LU>

```

```

483 {
484   \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
485 }
486 \</LU>

```

#### 1.4.2 Font transformations

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

```

487 \keys_define:nn {fontspec}
488 {
489   FakeSlant .code:n =
490   {
491     \@@_update_featstr:n{slant=#1}
492   },
493   FakeSlant .default:n = {0.2}
494 }
495 \keys_define:nn {fontspec}
496 {
497   FakeStretch .code:n =
498   {
499     \@@_update_featstr:n{extend=#1}
500   },
501   FakeStretch .default:n = {1.2}
502 }
503 \< *XE>
504 \keys_define:nn {fontspec}
505 {
506   FakeBold .code:n =
507   {
508     \@@_update_featstr:n {embolden=#1}
509   },
510   FakeBold .default:n = {1.5}
511 }
512 \</XE>
513 \< *LU>
514 \keys_define:nn {fontspec}
515 {
516   FakeBold .code:n = { \@@_warning:n {fakebold-only-xetex} }
517 }
518 \</LU>

```

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.)

```

519 \keys_define:nn {fontspec}
520 {
521   AutoFakeSlant .code:n =
522   {

```

```

523 \bool_if:NT \l_@@_firsttime_bool
524 {
525   \tl_set:Nn \l_@@_fake_slant_tl {#1}
526   \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
527   \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
528   \bool_set_false:N \l_@@_noit_bool
529
530   \tl_if_empty:NF \l_@@_fake_embolden_tl
531   {
532     \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
533       {FakeBold=\l_@@_fake_embolden_tl}
534     \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
535     \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
536   }
537 }
538 },
539 AutoFakeSlant .default:n = {0.2}
540 }

```

Same but reversed:

```

541 \keys_define:nn {fontspec}
542 {
543   AutoFakeBold .code:n =
544   {
545     \bool_if:NT \l_@@_firsttime_bool
546     {
547       \tl_set:Nn \l_@@_fake_embolden_tl {#1}
548       \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
549       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
550       \bool_set_false:N \l_@@_nobf_bool
551
552       \tl_if_empty:NF \l_@@_fake_slant_tl
553       {
554         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
555           {FakeSlant=\l_@@_fake_slant_tl}
556         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
557         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
558       }
559     }
560   },
561   AutoFakeBold .default:n = {1.5}
562 }

```

### 1.4.3 Raw feature string

This allows savvy X<sub>Y</sub>TeX-ers to input font features manually if they have already memorised the OpenType abbreviations and don't mind not having error checking.

```

563 \@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}
564 {
565   \@@_update_featstr:n {#1}
566 }
567 \@@_keys_define_code:nnn {fontspec-aat} {RawFeature}

```

```
568 {  
569   \@@_update_featstr:n {#1}  
570 }
```

## File XIV

# fontspec-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 / TeXReset .code:n = { \tl_clear:N \l_@@_mapping_tl },
24 }
25 </XE>
26 <LU>\@@_define_opentype_onreset:nnnnn {Ligatures} {TeX} {} { +tlig } {}
```

### 2.2 Letters

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

```

36 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercaseSmallCaps} {c2sc} {c2sc} {+c2pc,+uni
37 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercasePetiteCaps} {c2pc} {c2pc} {+c2sc,+un
38 \@@_define_opentype_onoffreset:nnnnn {Letters} {Uppercase} {unic} {unic} {+rand}
39 \@@_define_opentype_onoffreset:nnnnn {Letters} {Random} {rand} {rand} {+unic}

```

## 2.3 Numbers

```

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

```

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

```

## 2.4 Vertical position

```

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

```

## 2.5 Contextuals

```

75 \@@_define_opentype_feature_group:n {Contextuals}

```

```

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

```

## 2.6 Diacritics

```

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

```

## 2.7 Kerning

```

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

```

## 2.8 Fractions

```

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

```

```

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

```

## 2.9 Style

```

123 \@@_define_opentype_feature_group:n {Style}
124 \@@_define_opentype_feature:nnnnn {Style} {ResetAll} {} {}
125 {
126   +salt,-salt,
127   +ital,-ital,
128   +ruby,-ruby,
129   +swsh,-swsh,
130   +hist,-hist,
131   +titl,-titl,
132   +hkna,-hkna,
133   +vkna,-vkna,
134   +ssty=0,-ssty=0,
135   +ssty=1,-ssty=1,
136 }
137 \@@_define_opentype_onoffreset:nnnnn {Style} {Alternate} {salt} {salt} {}
138 \@@_define_opentype_onoffreset:nnnnn {Style} {Italic} {ital} {ital} {}
139 \@@_define_opentype_onoffreset:nnnnn {Style} {Ruby} {ruby} {ruby} {}
140 \@@_define_opentype_onoffreset:nnnnn {Style} {Swash} {swsh} {swsh} {}
141 \@@_define_opentype_onoffreset:nnnnn {Style} {Cursive} {swsh} {curs} {}
142 \@@_define_opentype_onoffreset:nnnnn {Style} {Historic} {hist} {hist} {}
143 \@@_define_opentype_onoffreset:nnnnn {Style} {TitlingCaps} {titl} {titl} {}
144 \@@_define_opentype_onoffreset:nnnnn {Style} {HorizontalKana} {hkna} {hkna} {+vkna,+pkna}
145 \@@_define_opentype_onoffreset:nnnnn {Style} {VerticalKana} {vkna} {vkna} {+hkna,+pkna}
146 \@@_define_opentype_onoffreset:nnnnn {Style} {ProportionalKana} {pkna} {pkna} {+vkna,+hkna}
147 \@@_define_opentype_feature:nnnnn {Style} {MathScript} {ssty} {+ssty=0} {+ssty=1}
148 \@@_define_opentype_feature:nnnnn {Style} {MathScriptScript} {ssty} {+ssty=1} {+ssty=0}

```

## 2.10 CJK shape

```

149 \@@_define_opentype_feature_group:n {CJKShape}
150 \@@_define_opentype_feature:nnnnn {CJKShape} {ResetAll} {} {}
151 {
152   +trad,-trad,
153   +smpl,-smpl,
154   +jp78,-jp78,
155   +jp83,-jp83,
156   +jp90,-jp90,
157   +jp04,-jp04,
158   +expt,-expt,
159   +nlck,-nlck,
160 }
161 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Traditional} {trad} {trad} {+smpl,+jp78,+jp83}
162 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Simplified} {smpl} {smpl} {+trad,+jp78,+jp83}
163 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978} {jp78} {jp78} {+trad,+smpl,+jp83}
164 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983} {jp83} {jp83} {+trad,+smpl,+jp78}
165 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990} {jp90} {jp90} {+trad,+smpl,+jp78}

```

```

166 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS2004} {jp04} {jp04} {+trad,+smpl,+jp7
167 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Expert} {expt} {expt} {+trad,+smpl,+jp7
168 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {NLC} {nlck} {nlck} {+trad,+smpl,+jp7

```

## 2.11 Character width

```

169 \@@_define_opentype_feature_group:n {CharacterWidth}
170 \@@_define_opentype_feature:nnnnn {CharacterWidth} {ResetAll} {} {}
171 {
172 +pwid,-pwid,
173 +fwid,-fwid,
174 +hwid,-hwid,
175 +twid,-twid,
176 +qwid,-qwid,
177 +palt,-palt,
178 +halt,-halt,
179 }
180 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Proportional} {pwid} {pwid} {
181 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Full} {fwid} {fwid} {
182 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Half} {hwid} {hwid} {
183 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Third} {twid} {twid} {
184 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Quarter} {qwid} {qwid} {
185 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateProportional} {palt} {palt} {
186 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateHalf} {halt} {halt} {

```

## 2.12 Vertical

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

```

187 \@@_define_opentype_feature_group:n {Vertical}
188 \@@_define_opentype_onoffreset:nnnnn {Vertical} {RotatedGlyphs} {vrt2} {vrt2} {+vrtr,
189 \@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternatesForRotation} {vrtr} {vrtr} {+vrt2}
190 \@@_define_opentype_onoffreset:nnnnn {Vertical} {Alternates} {vert} {vert} {+vrt2}
191 \@@_define_opentype_onoffreset:nnnnn {Vertical} {KanaAlternates} {vkna} {vkna} {+hkna}
192 \@@_define_opentype_onoffreset:nnnnn {Vertical} {Kerning} {vkern} {vkern} {}
193 \@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternateMetrics} {valt} {valt} {+vhal,
194 \@@_define_opentype_onoffreset:nnnnn {Vertical} {HalfMetrics} {vhal} {vhal} {+valt,
195 \@@_define_opentype_onoffreset:nnnnn {Vertical} {ProportionalMetrics} {vpal} {vpal} {+valt,

```

# 3 OpenType features that need numbering

## 3.1 Alternate

```

196 \@@_define_opentype_feature_group:n {Alternate}
197 \keys_define:nn {fontspec-opentype}
198 {
199 Alternate .default:n = {} ,
200 <LU> Alternate / Random .code:n =
201 <LU> { \@@_make_OT_feature:nnn {salt}{+salt = random }{} } ,
202 Alternate / unknown .code:n =
203 {

```

```

204     \clist_map_inline:nn {#1}
205     { \@@_make_OT_feature:nnn {salt}{ +salt = ##1 }{} }
206   }
207 }
208 \aliasfontfeature{Alternate}{StylisticAlternates}

```

### 3.2 Variant / StylisticSet

```

209 \@@_define_opentype_feature_group:n {Variant}
210 \keys_define:nn {fontspec-opentype}
211 {
212   Variant .default:n = {0} ,
213   Variant / unknown .code:n =
214   {
215     \clist_map_inline:nn {#1}
216     {
217       \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
218     }
219   }
220 }
221 \aliasfontfeature{Variant}{StylisticSet}

```

### 3.3 CharacterVariant

```

222 \@@_define_opentype_feature_group:n {CharacterVariant}
223 \use:x
224 {
225   \cs_new:Npn \exp_not:N \fontspec_parse_cv:w
226     ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
227   {
228     \@@_make_OT_feature:xxx
229     { cv \exp_not:N \two@digits {##1} } { +cv \exp_not:N \two@digits {##1} = ##2 } {}
230   }
231   \keys_define:nn {fontspec-opentype}
232   {
233     CharacterVariant / unknown .code:n =
234     {
235       \clist_map_inline:nn {##1}
236       {
237         \exp_not:N \fontspec_parse_cv:w
238         #####1 \c_colon_str 0 \c_colon_str \exp_not:N \q_nil
239       }
240     }
241   }
242 }

```

Possibilities: a:0:\q\_nil or a:b:0:\q\_nil.

### 3.4 Annotation

```

243 \@@_define_opentype_feature_group:n {Annotation}
244 \keys_define:nn {fontspec-opentype}
245 {
246   Annotation .default:n = {0} ,

```

```

247 Annotation / unknown .code:n =
248 {
249   \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
250 }
251 }

```

### 3.5 Ornament

```

252 \@@_define_opentype_feature_group:n {Ornament}
253 \keys_define:nn {fontspec-opentype}
254 {
255   Ornament .default:n = {0} ,
256   Ornament / unknown .code:n =
257   {
258     \@@_make_OT_feature:nnn {ornm} {+ornm=#1} {}
259   }
260 }

```

## 4 Script and Language

### 4.1 Script

```

261 \keys_define:nn { fontspec-opentype } { Script .choice: }
262 \cs_new:Nn \fontspec_new_script:nn
263 {
264   \keys_define:nn { fontspec-opentype } { Script / #1 .code:n =
265     \bool_set_false:N \l_@@_script_exist_bool
266     \clist_map_inline:nn {#2}
267     {
268       \@@_check_script:NnTF \l_fontspec_font {####1}
269       {
270         \tl_set:Nn \l_fontspec_script_tl {####1}
271         \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
272         \bool_set_true:N \l_@@_script_exist_bool
273         \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
274         \clist_map_break:
275       }
276     }
277   }
278   \bool_if:NF \l_@@_script_exist_bool
279   {
280     \str_if_eq:nnTF {#1} {Latin}
281     {
282       \@@_warning:nx {script-not-exist} {#1}
283     }
284     {
285       \@@_check_script:NnTF \l_fontspec_font {latn}
286       {
287         \@@_warning:nx {script-not-exist-latn} {#1}
288         \tl_set:Nn \l_fontspec_script_tl {latn}
289         \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
290       }
291       {

```

```

292         \@@_warning:nx {script-not-exist} {#1}
293     }
294 }
295 }
296 }
297 }

```

## 4.2 Language

```

298 \keys_define:nn { fontspec-opentype } { Language .choice: }
299 \cs_new:Nn \fontspec_new_lang:nn
300 {
301     \keys_define:nn { fontspec-opentype } { Language / #1 .code:n =
302         \@@_check_lang:NnTF \l_fontspec_font {#2}
303         {
304             \tl_set:Nn \l_fontspec_lang_tl {#2}
305             \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
306             \tl_gset:Nx \g_@@_single_feat_tl { language=#2 }
307         }
308         {
309             \@@_warning:nx {language-not-exist} {#1}
310             \keys_set:nn { fontspec-opentype } { Language = Default }
311         }
312     }
313 }

```

### Default

```

314 \@@_keys_define_code:nnn {fontspec-opentype}{ Language / Default }
315 {
316     \tl_set:Nn \l_fontspec_lang_tl {DFLT}
317     \int_zero:N \l_@@_language_int
318     \tl_gset:Nn \g_@@_single_feat_tl { language=DFLT }
319 }

```

**Turkish** Turns out that many fonts use ‘TUR’ as their Turkish language tag rather than the specified ‘TRK’. So we check for both:

```

320 \keys_define:nn {fontspec-opentype}
321 {
322     Language / Turkish .code:n =
323     {
324         \@@_check_lang:NnTF \l_fontspec_font {TRK}
325         {
326             \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
327             \tl_set:Nn \l_fontspec_lang_tl {TRK}
328             \tl_gset:Nn \g_@@_single_feat_tl { language=TRK }
329         }
330         {
331             \@@_check_lang:NnTF \l_fontspec_font {TUR}
332             {
333                 \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
334                 \tl_set:Nn \l_fontspec_lang_tl {TUR}
335                 \tl_gset:Nn \g_@@_single_feat_tl { language=TUR }

```

```

336     }
337     {
338         \@@_warning:nx {language-not-exist} {Turkish}
339         \keys_set:nn {fontspec-opentype} {Language=Default}
340     }
341 }
342 }
343 }

```

## 5 Backwards compatibility

Backwards compatibility:

```

344 \cs_new:Nn \@@_ot_compat:nn
345 {
346     \aliasfontfeatureoption {#1} {#20ff} {No#2}
347 }
348 \@@_ot_compat:nn {Ligatures} {Rare}
349 \@@_ot_compat:nn {Ligatures} {Required}
350 \@@_ot_compat:nn {Ligatures} {Common}
351 \@@_ot_compat:nn {Ligatures} {Discretionary}
352 \@@_ot_compat:nn {Ligatures} {Contextual}
353 \@@_ot_compat:nn {Ligatures} {Historic}
354 \@@_ot_compat:nn {Numbers} {SlashedZero}
355 \@@_ot_compat:nn {Contextuals} {Swash}
356 \@@_ot_compat:nn {Contextuals} {Alternate}
357 \@@_ot_compat:nn {Contextuals} {WordInitial}
358 \@@_ot_compat:nn {Contextuals} {WordFinal}
359 \@@_ot_compat:nn {Contextuals} {LineFinal}
360 \@@_ot_compat:nn {Contextuals} {Inner}
361 \@@_ot_compat:nn {Diacritics} {MarkToBase}
362 \@@_ot_compat:nn {Diacritics} {MarkToMark}
363 \@@_ot_compat:nn {Diacritics} {AboveBase}
364 \@@_ot_compat:nn {Diacritics} {BelowBase}

```

## File XV

# fontspec-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{CJK~Ideographic}{hani}
26 \newfontscript{Coptic}{copt}
27 \newfontscript{Cypriot~Syllabary}{cpri}
28 \newfontscript{Cyrillic}{cyr1}
29 \newfontscript{Default}{DFLT}
30 \newfontscript{Deseret}{dsrt}
31 \newfontscript{Devanagari}{dev2,deva}
32 \newfontscript{Duployan}{dupl}
33 \newfontscript{Egyptian~Hieroglyphs}{egyp}
34 \newfontscript{Elbasan}{elba}
35 \newfontscript{Ethiopic}{ethi}
36 \newfontscript{Georgian}{geor}
37 \newfontscript{Glagolitic}{glag}
38 \newfontscript{Gothic}{goth}
39 \newfontscript{Grantha}{gran}
40 \newfontscript{Greek}{grek}
41 \newfontscript{Gujarati}{gjr2,gujr}
42 \newfontscript{Gurmukhi}{gur2,guru}
43 \newfontscript{Hangul~Jamo}{jamo}
44 \newfontscript{Hangul}{hang}
```

```

45 \newfontscript{Hanunoo}{hano}
46 \newfontscript{Hatran}{hatr}
47 \newfontscript{Hebrew}{hebr}
48 \newfontscript{Hiragana~and~Katakana}{kana}
49 \newfontscript{Imperial~Aramaic}{armi}
50 \newfontscript{Inscriptional~Pahlavi}{phli}
51 \newfontscript{Inscriptional~Parthian}{prti}
52 \newfontscript{Javanese}{java}
53 \newfontscript{Kaithi}{kthi}
54 \newfontscript{Kannada}{knd2,knda}
55 \newfontscript{Kayah~Li}{kali}
56 \newfontscript{Kharosthi}{khar}
57 \newfontscript{Khmer}{khmr}
58 \newfontscript{Khojki}{khoj}
59 \newfontscript{Khudawadi}{sind}
60 \newfontscript{Lao}{lao~}
61 \newfontscript{Latin}{latn}
62 \newfontscript{Lepcha}{lepc}
63 \newfontscript{Limbu}{limb}
64 \newfontscript{Linear~A}{lina}
65 \newfontscript{Linear~B}{linb}
66 \newfontscript{Lisu}{lisu}
67 \newfontscript{Lycian}{lyci}
68 \newfontscript{Lydian}{lydi}
69 \newfontscript{Mahajani}{mahj}
70 \newfontscript{Malayalam}{mlm2,mlym}
71 \newfontscript{Mandaic}{mand}
72 \newfontscript{Manichaeen}{mani}
73 \newfontscript{Marchen}{marc}
74 \newfontscript{Math}{math}
75 \newfontscript{Meitei~Mayek}{mtei}
76 \newfontscript{Mende~Kikakui}{mend}
77 \newfontscript{Meroitic~Cursive}{merc}
78 \newfontscript{Meroitic~Hieroglyphs}{mero}
79 \newfontscript{Miao}{plrd}
80 \newfontscript{Modi}{modi}
81 \newfontscript{Mongolian}{mong}
82 \newfontscript{Mro}{mroo}
83 \newfontscript{Multani}{mult}
84 \newfontscript{Musical~Symbols}{musc}
85 \newfontscript{Myanmar}{mym2,mymr}
86 \newfontscript{N'Ko}{nko~}
87 \newfontscript{Nabataean}{nbat}
88 \newfontscript{Newa}{newa}
89 \newfontscript{Odia}{ory2,orya}
90 \newfontscript{Ogham}{ogam}
91 \newfontscript{Ol~Chiki}{olck}
92 \newfontscript{Old~Italic}{ital}
93 \newfontscript{Old~Hungarian}{hung}
94 \newfontscript{Old~North~Arabian}{narb}
95 \newfontscript{Old~Permic}{perm}

```

```

96 \newfontscript{Old-Persian~Cuneiform}{xpeo}
97 \newfontscript{Old-South-Arabian}{sarb}
98 \newfontscript{Old-Turkic}{orkh}
99 \newfontscript{Osage}{osge}
100 \newfontscript{Osmanya}{osma}
101 \newfontscript{Pahawh~Hmong}{hmng}
102 \newfontscript{Palmyrene}{palm}
103 \newfontscript{Pau-Cin-Hau}{pauc}
104 \newfontscript{Phags-pa}{phag}
105 \newfontscript{Phoenician}{phnx}
106 \newfontscript{Psalter~Pahlavi}{phlp}
107 \newfontscript{Rejang}{rjng}
108 \newfontscript{Runic}{runr}
109 \newfontscript{Samaritan}{samr}
110 \newfontscript{Saurashtra}{saur}
111 \newfontscript{Sharada}{shrd}
112 \newfontscript{Shavian}{shaw}
113 \newfontscript{Siddham}{sidd}
114 \newfontscript{Sign-Writing}{sgnw}
115 \newfontscript{Sinhala}{sinh}
116 \newfontscript{Sora-Sompeng}{sora}
117 \newfontscript{Sumero-Akkadian~Cuneiform}{xsux}
118 \newfontscript{Sundanese}{sund}
119 \newfontscript{Syloti-Nagri}{sylo}
120 \newfontscript{Syriac}{syr}
121 \newfontscript{Tagalog}{tglg}
122 \newfontscript{Tagbanwa}{tagb}
123 \newfontscript{Tai~Le}{tale}
124 \newfontscript{Tai~Lu}{tal}
125 \newfontscript{Tai~Tham}{lana}
126 \newfontscript{Tai-Viet}{tvt}
127 \newfontscript{Takri}{takr}
128 \newfontscript{Tamil}{tml2,taml}
129 \newfontscript{Tangut}{tang}
130 \newfontscript{Telugu}{tel2,telu}
131 \newfontscript{Thaana}{thaa}
132 \newfontscript{Thai}{thai}
133 \newfontscript{Tibetan}{tib}
134 \newfontscript{Tifinagh}{tfng}
135 \newfontscript{Tirhuta}{tirh}
136 \newfontscript{Ugaritic~Cuneiform}{ugar}
137 \newfontscript{Vai}{vai}
138 \newfontscript{Warang-Citi}{wara}
139 \newfontscript{Yi}{yi}

```

For convenience or backwards compatibility:

```

140 \newfontscript{CJK}{h}
141 \newfontscript{Kana}{kana}
142 \newfontscript{Maths}{math}
143 \newfontscript{N'ko}{nko}
144 \newfontscript{Oriya}{ory2,orya}

```

## File XVI

# fontspec-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}

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

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

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

249 \newfontlanguage{Naga-Assamese}{NAG}  
 250 \newfontlanguage{Nanai}{NAN}  
 251 \newfontlanguage{Naskapi}{NAS}  
 252 \newfontlanguage{N-Cree}{NCR}  
 253 \newfontlanguage{Ndebele}{NDB}  
 254 \newfontlanguage{Ndonga}{NDG}  
 255 \newfontlanguage{Nepali}{NEP}  
 256 \newfontlanguage{Newari}{NEW}  
 257 \newfontlanguage{Nagari}{NGR}  
 258 \newfontlanguage{Norway-House-Cree}{NHC}  
 259 \newfontlanguage{Nisi}{NIS}  
 260 \newfontlanguage{Niuean}{NIU}  
 261 \newfontlanguage{Nkole}{NKL}  
 262 \newfontlanguage{N'ko}{NKO}  
 263 \newfontlanguage{Dutch}{NLD}  
 264 \newfontlanguage{Nogai}{NOG}  
 265 \newfontlanguage{Norwegian}{NOR}  
 266 \newfontlanguage{Northern-Sami}{NSM}  
 267 \newfontlanguage{Northern-Tai}{NTA}  
 268 \newfontlanguage{Esperanto}{NTO}  
 269 \newfontlanguage{Nynorsk}{NYN}  
 270 \newfontlanguage{Oji-Cree}{OCR}  
 271 \newfontlanguage{Ojibway}{OBJ}  
 272 \newfontlanguage{Oriya}{ORI}  
 273 \newfontlanguage{Oromo}{ORO}  
 274 \newfontlanguage{Ossetian}{OSS}  
 275 \newfontlanguage{Palestinian-Aramaic}{PAA}  
 276 \newfontlanguage{Pali}{PAL}  
 277 \newfontlanguage{Punjabi}{PAN}  
 278 \newfontlanguage{Palpa}{PAP}  
 279 \newfontlanguage{Pashto}{PAS}  
 280 \newfontlanguage{Polytonic-Greek}{PGR}  
 281 \newfontlanguage{Pilipino}{PIL}  
 282 \newfontlanguage{Palaung}{PLG}  
 283 \newfontlanguage{Polish}{PLK}  
 284 \newfontlanguage{Provençal}{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{Turkmen}{TKM}  
 347 \newfontlanguage{Temne}{TMN}  
 348 \newfontlanguage{Tswana}{TNA}  
 349 \newfontlanguage{Tundra~Nenets}{TNE}  
 350 \newfontlanguage{Tonga}{TNG}

```

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

```

## File XVII

# fontspec-feat-aat.dtx

## 1 AAT feature definitions

These are only defined for X<sub>Y</sub>TeX.

### 1.1 Ligatures

```
1 \@@_define_aat_feature_group:n {Ligatures}
2 \@@_define_aat_feature:nnnn {Ligatures} {Required} {1} {0}
3 \@@_define_aat_feature:nnnn {Ligatures} {NoRequired} {1} {1}
4 \@@_define_aat_feature:nnnn {Ligatures} {Common} {1} {2}
5 \@@_define_aat_feature:nnnn {Ligatures} {NoCommon} {1} {3}
6 \@@_define_aat_feature:nnnn {Ligatures} {Rare} {1} {4}
7 \@@_define_aat_feature:nnnn {Ligatures} {NoRare} {1} {5}
8 \@@_define_aat_feature:nnnn {Ligatures} {Discretionary} {1} {4}
9 \@@_define_aat_feature:nnnn {Ligatures} {NoDiscretionary} {1} {5}
10 \@@_define_aat_feature:nnnn {Ligatures} {Logos} {1} {6}
11 \@@_define_aat_feature:nnnn {Ligatures} {NoLogos} {1} {7}
12 \@@_define_aat_feature:nnnn {Ligatures} {Rebus} {1} {8}
13 \@@_define_aat_feature:nnnn {Ligatures} {NoRebus} {1} {9}
14 \@@_define_aat_feature:nnnn {Ligatures} {Diphthong} {1} {10}
15 \@@_define_aat_feature:nnnn {Ligatures} {NoDiphthong} {1} {11}
16 \@@_define_aat_feature:nnnn {Ligatures} {Squared} {1} {12}
17 \@@_define_aat_feature:nnnn {Ligatures} {NoSquared} {1} {13}
18 \@@_define_aat_feature:nnnn {Ligatures} {AbbrevSquared} {1} {14}
19 \@@_define_aat_feature:nnnn {Ligatures} {NoAbbrevSquared} {1} {15}
20 \@@_define_aat_feature:nnnn {Ligatures} {Icelandic} {1} {32}
21 \@@_define_aat_feature:nnnn {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:nnnn {Letters} {Normal} {3} {0}
31 \@@_define_aat_feature:nnnn {Letters} {Uppercase} {3} {1}
32 \@@_define_aat_feature:nnnn {Letters} {Lowercase} {3} {2}
33 \@@_define_aat_feature:nnnn {Letters} {SmallCaps} {3} {3}
34 \@@_define_aat_feature:nnnn {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:nnnn {Numbers} {Monospaced} {6} {0}
37 \@@_define_aat_feature:nnnn {Numbers} {Proportional} {6} {1}
38 \@@_define_aat_feature:nnnn {Numbers} {Lowercase} {21} {0}
39 \@@_define_aat_feature:nnnn {Numbers} {OldStyle} {21} {0}
40 \@@_define_aat_feature:nnnn {Numbers} {Uppercase} {21} {1}
41 \@@_define_aat_feature:nnnn {Numbers} {Lining} {21} {1}
42 \@@_define_aat_feature:nnnn {Numbers} {SlashedZero} {14} {5}
43 \@@_define_aat_feature:nnnn {Numbers} {NoSlashedZero} {14} {4}
```

### 1.4 Contextuals

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

### 1.5 Diacritics

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

### 1.6 Vertical position

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

### 1.7 Fractions

```
64 \@@_define_aat_feature_group:n {Fractions}
65 \@@_define_aat_feature:nnnn {Fractions} {On} {11} {1}
66 \@@_define_aat_feature:nnnn {Fractions} {Off} {11} {0}
67 \@@_define_aat_feature:nnnn {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 }
100

```

## 1.10 Style

```

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

```

## 1.11 CJK shape

```

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

```

```

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

```

## 1.12 Character width

```

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

```

## 1.13 Annotation

```

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

```

## File XVIII

# fontspec-enc.dtx

## 1 Extended font encodings

To be removed after the 2017 release of LaTeX2e:

```
1 \providecommand\UnicodeFontFile[2]{"[#1]:#2"}
2 \providecommand\UnicodeFontName[2]{"#1:#2"}
3 \langle XE \rangle \providecommand\UnicodeFontTeXLigatures{mapping=tex-text;}
4 \langle LU \rangle \providecommand\UnicodeFontTeXLigatures{+tlig;}

5 \providecommand\add@unicode@accent[2]{#2\char#1\relax}
6 \providecommand\DeclareUnicodeAccent[3]{%
7   \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}}%
8 }
```

`\EncodingCommand`

```
9 \DeclareDocumentCommand \EncodingCommand {m0{m}}
10 {
11   \bool_if:NF \l_@@_defining_encoding_bool
12   { \@@_error:nn {only-inside-encdef} \EncodingCommand }
13   \DeclareTextCommand{#1}{\UnicodeEncodingName}{#2}{#3}
14 }
```

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

`\EncodingAccent`

```
15 \DeclareDocumentCommand \EncodingAccent {mm}
16 {
17   \bool_if:NF \l_@@_defining_encoding_bool
18   { \@@_error:nn {only-inside-encdef} \EncodingAccent }
19   \DeclareTextCommand{#1}{\UnicodeEncodingName}{\add@unicode@accent{#2}}
20 }
```

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

`\EncodingSymbol`

```
21 \DeclareDocumentCommand \EncodingSymbol {mm}
22 {
23   \bool_if:NF \l_@@_defining_encoding_bool
24   { \@@_error:nn {only-inside-encdef} \EncodingSymbol }
25   \DeclareTextSymbol{#1}{\UnicodeEncodingName}{#2}
26 }
```

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

`\EncodingComposite`

```
27 \DeclareDocumentCommand \EncodingComposite {mmm}
28 {
29   \bool_if:NF \l_@@_defining_encoding_bool
30   { \@@_error:nn {only-inside-encdef} \EncodingComposite }
```

```

31 \DeclareTextComposite{#1}{\UnicodeEncodingName}{#2}{#3}
32 }

```

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

\EncodingCompositeCommand

```

33 \DeclareDocumentCommand \EncodingCompositeCommand {mmm}
34 {
35   \bool_if:NF \l_@@_defining_encoding_bool
36   { \@@_error:nn {only-inside-encdef} \EncodingCompositeCommand }
37   \DeclareTextCompositeCommand{#1}{\UnicodeEncodingName}{#2}{#3}
38 }

```

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

\DeclareUnicodeEncoding

```

39 \DeclareDocumentCommand \DeclareUnicodeEncoding {mm}
40 {
41   \DeclareFontEncoding{#1}{}{}
42   \DeclareErrorFont{#1}{lmr}{m}{n}{10}
43   \DeclareFontSubstitution{#1}{lmr}{m}{n}
44   \DeclareFontFamily{#1}{lmr}{}
45
46   \DeclareFontShape{#1}{lmr}{m}{n}
47     {<->\UnicodeFontFile{lmroman10-regular}{\UnicodeFontTeXLigatures}}{}
48   \DeclareFontShape{#1}{lmr}{m}{it}
49     {<->\UnicodeFontFile{lmroman10-italic}{\UnicodeFontTeXLigatures}}{}
50   \DeclareFontShape{#1}{lmr}{m}{sc}
51     {<->\UnicodeFontFile{lmromancaps10-regular}{\UnicodeFontTeXLigatures}}{}
52   \DeclareFontShape{#1}{lmr}{bx}{n}
53     {<->\UnicodeFontFile{lmroman10-bold}{\UnicodeFontTeXLigatures}}{}
54   \DeclareFontShape{#1}{lmr}{bx}{it}
55     {<->\UnicodeFontFile{lmroman10-bolditalic}{\UnicodeFontTeXLigatures}}{}
56
57   \tl_set_eq:NN \l_@@_prev_unicode_name_tl \UnicodeEncodingName
58   \tl_set:Nn \UnicodeEncodingName {#1}
59   \bool_set_true:N \l_@@_defining_encoding_bool
60   #2
61   \bool_set_false:N \l_@@_defining_encoding_bool
62   \tl_set_eq:NN \UnicodeEncodingName \l_@@_prev_unicode_name_tl
63 }

```

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

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

\UndeclareAccent      64 \DeclareDocumentCommand \UndeclareSymbol {m}

```

65 {
66   \bool_if:NF \l_@@_defining_encoding_bool
67   { \@@_error:nn {only-inside-encdef} \UndeclareSymbol }
68   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
69 }
70 \DeclareDocumentCommand \UndeclareAccent {m}

```

```

71 {
72   \bool_if:NF \l_@@_defining_encoding_bool
73   { \@@_error:nn {only-inside-encdef} \UndeclareAccent }
74   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
75 }
76 \DeclareDocumentCommand \UndeclareCommand {m}
77 {
78   \bool_if:NF \l_@@_defining_encoding_bool
79   { \@@_error:nn {only-inside-encdef} \UndeclareCommand }
80   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
81 }

```

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

## **\UndeclareComposite**

```

82 \DeclareDocumentCommand \UndeclareComposite {mm}
83 {
84   \bool_if:NF \l_@@_defining_encoding_bool
85   { \@@_error:nn {only-inside-encdef} \UndeclareComposite }
86   \cs_undefine:c
87   { \c_backslash_str \UnicodeEncodingName \token_to_str:N #1 - \tl_to_str:n {#2} }
88 }

```

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

## File XIX

# fontspec-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  $\TeX$ 's `operators` maths font to still go back to the legacy `cmr` font for all these random glyphs, unless a separate maths font package has been loaded instead.

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

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

```

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

```

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

```

% % fontmath.ltx:
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}
% \DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
%
% % amsmath.sty:
% \renewcommand{\colon}{\nobreak\mskip2mu\mathpunct{}\nonscript
% \mkern-\thinmuskip{:}\mskip6mu\plus1mu\relax}
%
% % euler.sty:
% \DeclareMathSymbol{:}{\mathrel}{EulerFraktur}{"3A}
%
% % lucbmath.sty:
% \DeclareMathSymbol{@tempb}{\mathpunct}{operators}{58}
% \ifx\colon@tempb
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{58}
% \fi
% \DeclareMathSymbol{:}{\mathrel}{operators}{58}

```

(3A<sub>16</sub> = 58<sub>10</sub>) 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 }
72 }

```

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  $\TeX$  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`.

```

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

```

```

89   }
90   \SetMathAlphabet\mathsf{bold}\g_fontspec_encoding_tl\g_@@_mathsf_tl\bfdefault\updefault
91   \SetMathAlphabet\mathtt{bold}\g_fontspec_encoding_tl\g_@@_mathtt_tl\bfdefault\updefault
92   }

```

(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<sup>A</sup>T<sub>E</sub>X Font Catalogue': <http://www.tug.dk/FontCatalogue/mathfonts.html>. I'm sure there are more I've missed. Do the T<sub>E</sub>X 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.

```

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

```

*(End definition for \fontspec\_maybe\_setup\_maths:. This function is documented on page ??.)*

File XX

## fontspec-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-xfss.dtx

## 1 Changes to the NFSS

1 `<*fontspec>`

### 1.1 Italic small caps and so on

`\sishape` These commands for actually selecting italic small caps have been defined for many years;  
`\textsi` I'm inclined to drop them. They're probably used very infrequently; I personally prefer just writing `\textit{\textsc{...}}` instead.

```
2 \providecommand*\itscdefault{\itdefault\scdefault}
3 \providecommand*\slscdefault{\sldefault\scdefault}
4 \DeclareRobustCommand{\sishape}
5 {
6   \not@math@alphabet\sishape\relax
7   \fontshape{\itscdefault}\selectfont
8 }
9 \DeclareTextFontCommand{\textsi}{\sishape}
```

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

ℒ<sub>TEX</sub>'s 'shape' font axis needs to be overloaded to support italic small caps and slanted small caps. These are the combinations to support:

```
10 \cs_new:Nn \@@_shape_merge:nn { c_@@_shape_#1_#2_tl }
11 \tl_const:cn { \@@_shape_merge:nn \itdefault \scdefault } {\itscdefault}
12 \tl_const:cn { \@@_shape_merge:nn \sldefault \scdefault } {\slscdefault}
13 \tl_const:cn { \@@_shape_merge:nn \scdefault \itdefault } {\itscdefault}
14 \tl_const:cn { \@@_shape_merge:nn \scdefault \sldefault } {\slscdefault}
15 \tl_const:cn { \@@_shape_merge:nn \slscdefault \itdefault } {\itscdefault}
16 \tl_const:cn { \@@_shape_merge:nn \itscdefault \sldefault } {\slscdefault}
17 \tl_const:cn { \@@_shape_merge:nn \itscdefault \updefault } {\scdefault}
18 \tl_const:cn { \@@_shape_merge:nn \slscdefault \updefault } {\scdefault}
```

`\fontspec_merge_shape:n` These macros enable the overload on the `\. .shape` commands. First, a shape 'new+current' (prefix) or 'current+new' (suffix) is tried. If not found, fall back on the 'new' shape.

```
19 \cs_new:Nn \fontspec_merge_shape:n
20 {
21   \@@_if_merge_shape:nTF {#1}
22     { \fontshape { \tl_use:c { \@@_shape_merge:nn {\f@shape} {#1} } } \selectfont }
23     { \fontshape {#1} \selectfont }
24 }
```

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.

```
25 \prg_new_conditional:Nnn \@@_if_merge_shape:n {TF}
26 {
27   \bool_lazy_and:nnTF
28     { \tl_if_exist_p:c { \@@_shape_merge:nn {\f@shape} {#1} } }
29     {
```

```

30     \cs_if_exist_p:c
31     {
32         \f@encoding/\f@family/\f@series/
33         \tl_use:c { \@@_shape_merge:nn {\f@shape} {#1} }
34     }
35 }
36 \prg_return_true: \prg_return_false:
37 }

```

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

`\itshape` The original `\. .shape` commands are redefined to use the merge shape macro.

```

\scshape 38 \DeclareRobustCommand \itshape
\upshape 39 {
\slshape 40     \not@math@alphabet\itshape\mathit
41     \fontspec_merge_shape:n\itdefault
42 }
43 \DeclareRobustCommand \slshape
44 {
45     \not@math@alphabet\slshape\relax
46     \fontspec_merge_shape:n\sldefault
47 }
48 \DeclareRobustCommand \scshape
49 {
50     \not@math@alphabet\scshape\relax
51     \fontspec_merge_shape:n\scdefault
52 }
53 \DeclareRobustCommand \upshape
54 {
55     \not@math@alphabet\upshape\relax
56     \fontspec_merge_shape:n\updefault
57 }

```

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

## 1.2 Emphasis

`\emfontdeclare`

```

58 \cs_new_protected:Npn \emfontdeclare #1
59 {
60     \prop_gclear:N \g_@@_em_prop
61     \int_zero:N \l_@@_emdef_int
62     \bool_gset_true:N \g_@@_em_normalise_slant_bool
63
64     \tl_if_in:nnT {#1} {\slshape}
65     {
66         \tl_if_in:nnT {#1} {\itshape}
67         {
68             \bool_gset_false:N \g_@@_em_normalise_slant_bool
69         }
70     }
71 }

```

```

72 \group_begin:
73 \normalfont
74 \clist_map_inline:nn {\emreset,#1}
75 {
76   ##1
77   \prop_gput_if_new:NxV \g_@@_em_prop { \f@shape } { \l_@@_emdef_int }
78   \prop_gput:Nxn \g_@@_em_prop { switch-\int_use:N \l_@@_emdef_int } { ##1 }
79   \int_incr:N \l_@@_emdef_int
80 }
81 \group_end:
82 }

```

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

`\em`

```

83 \DeclareRobustCommand \em
84 {
85   \@nomath\em
86   \tl_set:Nx \l_@@_emshape_query_tl { \f@shape }
87
88   \bool_if:NT \g_@@_em_normalise_slant_bool
89   {
90     \tl_replace_all:Nnn \l_@@_emshape_query_tl {/sl} {/it}
91   }
92
93   <debug> \typeout{Emph~ level:~\int_use:N \l_@@_em_int}
94   \prop_get:NxNT \g_@@_em_prop { \l_@@_emshape_query_tl } \l_@@_em_tmp_tl
95   {
96     \int_set:Nn \l_@@_em_int { \l_@@_em_tmp_tl }
97   <debug> \typeout{Shape~ (\l_@@_emshape_query_tl)~ detected;~ new~ level:~\int_use:N \l_@@_em_int}
98   }
99
100   \int_incr:N \l_@@_em_int
101
102   \prop_get:NxNTF \g_@@_em_prop { switch-\int_use:N \l_@@_em_int } \l_@@_em_switch_tl
103   { \l_@@_em_switch_tl }
104   {
105     \int_zero:N \l_@@_em_int
106     \emreset
107   }
108
109 }

```

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

```

\emph
\emshape 110 \DeclareTextFontCommand{\emph}{\em}
\eminnershape 111 \cs_set:Npn \emreset { \upshape }
\emreset 112 \cs_set:Npn \emshape { \itshape }
113 \cs_set:Npn \eminnershape { \upshape }

```

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

### 1.3 Strong emphasis

`\strongfontdeclare`

```

114 \cs_new_protected:Npn \strongfontdeclare #1
115 {
116   \prop_gclear:N \g_@@_strong_prop
117   \int_zero:N \l_@@_strongdef_int
118
119   \group_begin:
120     \normalfont
121     \clist_map_inline:nn {\strongreset,#1}
122     {
123       ##1
124       \prop_gput_if_new:NxV \g_@@_strong_prop { \f@series } { \l_@@_strongdef_int }
125       \prop_gput:Nxn \g_@@_strong_prop { switch-\int_use:N \l_@@_strongdef_int } { ##1 }
126       \int_incr:N \l_@@_strongdef_int
127     }
128   \group_end:
129 }

```

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

`\strongenv`

```

130 \DeclareRobustCommand \strongenv
131 {
132   \@nomath\strongenv
133
134   <debug> \typeout{Strong~ level:~\int_use:N \l_@@_strong_int}
135   \prop_get:NxNT \g_@@_strong_prop { \f@series } \l_@@_strong_tmp_tl
136   {
137     \int_set:Nn \l_@@_strong_int { \l_@@_strong_tmp_tl }
138   <debug> \typeout{Series~ (\f@series)~ detected;~ new~ level:~\int_use:N \l_@@_strong_int}
139   }
140
141   \int_incr:N \l_@@_strong_int
142
143   \prop_get:NxNTF \g_@@_strong_prop { switch-\int_use:N \l_@@_strong_int } \l_@@_strong_sw
144   { \l_@@_strong_switch_tl }
145   {
146     \int_zero:N \l_@@_strong_int
147     \strongreset
148   }
149
150 }

```

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

`\strong`  
`\strongreset`

```

151 \DeclareTextFontCommand{\strong}{\strongenv}
152 \cs_set:Npn \strongreset {}

```

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

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

```
153 \cs_set:Npn \reset@font
154 {
155   \normalfont
156   \int_zero:N \l_@@_em_int
157   \int_zero:N \l_@@_strong_int
158 }
```

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

Programmer's interface for setting nesting levels:

```
159 \cs_new:Nn \fontspec_set_em_level:n { \int_set:Nn \l_@@_em_int {#1} }
160 \cs_new:Nn \fontspec_set_strong_level:n { \int_set:Nn \l_@@_strong_int {#1} }
```

Defaults:

```
161 \strongfontdeclare{ \bfseries }
162 \emfontdeclare{ \emshape, \eminnershape }
163 </fontspec>
```

## File XXII

# fontspec-patches.dtx

## 1 Patching code

1 `<*fontspec>`

### 1.1 `\-`

`\-` This macro is courtesy of Frank Mittelbach and the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> source code.

```
2 \DeclareRobustCommand{\-}{
3 {
4 \discretionary
5 {
6 \char\ifnum\hyphenchar\font<\z@
7 \xlx@defaultthyphenchar
8 \else
9 \hyphenchar\font
10 \fi
11 }{}{}
12 }
13 \def\xlx@defaultthyphenchar{`\-}
```

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

### 1.2 Verbatims

Many thanks to Apostolos Syropoulos for discovering this problem and writing the redefinition of L<sup>A</sup>T<sub>E</sub>X's `verbatim` environment and `\verb*` command.

`\fontspec_visible_space:` Print U+2423: OPEN BOX, which is used to visibly display a space character.

```
14 \cs_new:Nn \fontspec_visible_space:
15 {
16 \@@_primitive_font_glyph_if_exist:NnTF \font {"2423}
17 { \char"2423\scan_stop: }
18 { \fontspec_visible_space_fallback: }
19 }
```

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

`\fontspec_visible_space_fallback:` If the current font doesn't have U+2423: OPEN BOX, use Latin Modern Mono instead.

```
20 \cs_new:Nn \fontspec_visible_space_fallback:
21 {
22 {
23 \usefont{\g_fontspec_encoding_tl}{lmtt}{\f@series}{\f@shape}
24 \textvisiblespace
25 }
26 }
```

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

`\fontspec_print_visible_spaces:` Helper macro to turn spaces ( $\sim 20$ ) active and print visible space instead.

```

27 \group_begin:
28 \char_set_catcode_active:n{"20}%
29 \cs_gset:Npn\fontspec_print_visible_spaces:{%
30 \char_set_catcode_active:n{"20}%
31 \cs_set_eq:NN\sim\fontspec_visible_space:%
32 }%
33 \group_end:

```

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

`\verb` Redefine `\verb` to use `\fontspec_print_visible_spaces`:.  
`\verb*`

```

34 \def\verb
35 {
36   \relax\ifmmode\hbox\else\leavevmode\null\fi
37   \bgroup
38   \verb@eol@error \let\do\@makeother \dospecials
39   \verbatim@font\@noligs
40   \@ifstar\@sverb\@verb
41 }
42 \def\@sverb{\fontspec_print_visible_spaces:\@sverb}

```

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

It's better to put small things into `\AtBeginDocument`, so here we go:

```

43 \AtBeginDocument
44 {
45   \fontspec_patch_verbatim:
46   \fontspec_patch_moreverb:
47   \fontspec_patch_fancyvrb:
48   \fontspec_patch_listings:
49 }

```

`verbatim*` With the `verbatim` package.

```

50 \cs_set:Npn \fontspec_patch_verbatim:
51 {
52   \@ifpackageloaded{verbatim}
53   {
54     \cs_set:cpn {verbatim*}
55     {
56       \group_begin: \@verbatim \fontspec_print_visible_spaces: \verbatim@start
57     }
58   }

```

This is for vanilla  $\text{\LaTeX}$ .

```

59 {
60   \cs_set:cpn {verbatim*}
61   {
62     \@verbatim \fontspec_print_visible_spaces: \sxverbatim
63   }
64 }
65 }

```

`listingcont*` This is for `moreverb`. The main `listing*` environment inherits this definition.

```
66 \cs_set:Npn \fontspec_patch_moreverb:
67 {
68   \@ifpackageloaded{moreverb}{
69     \cs_set:cpn {listingcont*}
70     {
71       \cs_set:Npn \verbatim@processline
72       {
73         \thelisting@line \global\advance\listing@line\c_one
74         \the\verbatim@line\par
75       }
76       \verbatim \fontspec_print_visible_spaces: \verbatim@start
77     }
78   }{}
79 }
```

`listings` and `fancvrb` make things nice and easy:

```
80 \cs_set:Npn \fontspec_patch_fancyvrb:
81 {
82   \@ifpackageloaded{fancyvrb}
83   {
84     \cs_set_eq:NN \FancyVerbSpace \fontspec_visible_space:
85   }{}
86 }
87 \cs_set:Npn \fontspec_patch_listings:
88 {
89   \@ifpackageloaded{listings}
90   {
91     \cs_set_eq:NN \lst@visible_space \fontspec_visible_space:
92   }{}
93 }
```

### 1.3 `\oldstylenums`

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

`\liningnums`

```
94 \RenewDocumentCommand \oldstylenums {m}
95 {
96   { \addfontfeature{Numbers=OldStyle} #1 }
97 }
98 \NewDocumentCommand \liningnums {m}
99 {
100   { \addfontfeature{Numbers=Lining} #1 }
101 }
```

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

```
102 </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	
\#	221, 222, 253
\,	1, 2, 3, 4, 5, 17
\-	2, 4, 122, 648
@@ commands:	
\@@_DeclareFontShape:nnnnnn	501, 508, 519, 537
\g_@@_OT_features_prop	9, 11, 68
\@@_add_nfssfont:nnnn	259, 318, 319, 320, 321, 322, 323, 338
\@@_aff_error:n	11, 313, 354, 386
\l_@@_alias_bool	21, 204, 211, 217, 222, 229, 249
\l_@@_all_features_clist	21, 50, 99, 109, 123, 187, 284
\g_@@_all_keyval_modules_clist	1, 44, 206, 224
\g_@@_all_opentype_feature_names_prop	69, 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
\l_@@_arg_clist	56, 246, 247, 248, 251, 254
\l_@@_atsui_bool	7, 10, 217, 357, 364, 609
\l_@@_basename_tl	10, 40, 87, 335
\l_@@_bf_series_seq	42, 122, 134, 137
\g_@@_bfmathrm_tl	69, 70, 81, 86, 87, 88, 115
\@@_calc_scale:n	266, 267, 272
\g_@@_cfg_bool	1, 7, 8, 16
\l_@@_check_bool	5, 65, 66, 162, 167, 173, 189
\l_@@_check_feat_bool	28, 55, 57, 58
\@@_check_lang:Nn	111
\@@_check_lang:NnTF	99, 111, 114, 302, 324, 331
\@@_check_ot_feat:Nn	147
\@@_check_ot_feat:NnTF	50, 60, 69, 147, 601
\@@_check_script:Nn	79
\@@_check_script:NnTF	79, 82, 176, 268, 285
\@@_combo_sc_shape:n	509, 513, 558, 566
\@@_construct_font_call:nn	135, 137, 140, 142, 145, 169, 287, 399, 400, 421, 495
\@@_construct_font_call:nnnnnn	145, 154
\l_@@_curr_bfname_tl	89, 132, 142, 145, 147, 179
\l_@@_curr_fontname_tl	88, 329, 330
\g_@@_curr_series_tl	82, 121, 136, 140, 145, 147, 179, 644
\@@_declare_shape:nnnn	410, 429
\@@_declare_shape_loginfo:nn	441, 541
\@@_declare_shape_slanted:nn	440, 529
\@@_declare_shapes_normal:nn	438, 499
\@@_declare_shapes_smcaps:nn	439, 504
\g_@@_default_fontopts_clist	43, 111, 119
\@@_define_aat_feature:nnnn	2, 3, 3, 4, 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, 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, 183
\@@_define_aat_feature_group:n	1, 1, 29, 35, 44, 55, 59, 64, 68, 80, 91, 101, 108, 116, 125, 178

\@@_define_opentype_feature:nnnnn	\g_@@_family_int_prop ... 73, 261, 267
.. 5, 7, 28, 41, 41, 42, 43, 47, 48, 60,	\l_@@_family_label_tl .....
76, 92, 104, 110, 111, 112, 114, 119,	..... 105, 107, 121, 152, 159
120, 121, 124, 147, 148, 150, 170, 193	\@@_feat_off:n .....
\@@_define_opentype_feature_-	..... 37, 42
group:n .....	\@@_feat_prop_add:nn . 1, 2, 3, 4, 5, 5, 17
1, 6,	\@@_feat_reset:n .....
27, 40, 59, 75, 91, 103, 113, 123, 149,	..... 38, 43, 48
169, 187, 188, 196, 209, 222, 243, 252	\@@_find_autofonts: .....
\@@_define_opentype_onoffreset:nnnnn	..... 275, 295
.....	\l_@@_firsttime_bool .....
13, 14, 15, 16, 17, 18,	..... 1, 29, 172, 199, 271, 310,
33, 34, 35, 36, 37, 37, 38, 39, 50, 51,	400, 411, 423, 477, 523, 545, 617, 637
52, 53, 54, 58, 69, 70, 71, 72, 73, 74,	\@@_font_is_file: .....
85, 86, 87, 88, 89, 90, 99, 100, 101,	..... 30, 162, 168
102, 109, 122, 137, 138, 139, 140,	\@@_font_is_name: .....
141, 142, 143, 144, 145, 146, 161,	..... 162, 638
162, 163, 164, 165, 166, 167, 168,	\l_@@_font_path_tl ...
180, 181, 182, 183, 184, 185, 186,	..... 29, 92, 168, 639
188, 189, 190, 191, 192, 193, 194, 195	\@@_font_suppress_not_found_-
\@@_define_opentype_onreset:nnnnn	error: .....
.....	..... 5, 9, 24, 267
26, 45	\l_@@_fontcfg_bool ...
\g_@@_defined_shapes_tl .....	..... 11, 12, 18, 22, 81
.....	\l_@@_fontfeat_bf_clist .....
83, 190, 543, 643	..... 59, 176, 319, 548
\l_@@_defining_encoding_bool . 11,	\l_@@_fontfeat_bfit_clist .....
17, 23, 24, 29, 35, 59, 61, 66, 72, 78, 84	..... 61, 187, 322, 532, 534, 554, 556
\l_@@_disable_defaults_bool 20, 97, 153	\l_@@_fontfeat_bfsl_clist 63, 195, 323
\l_@@_em_int .....	\l_@@_fontfeat_clist 54, 130, 204, 272
.....	\l_@@_fontfeat_curr_clist .....
33, 93, 96, 97, 100, 102, 105, 156, 159	..... 55, 460, 469, 482
\g_@@_em_normalise_slant_bool ..	\l_@@_fontfeat_it_clist .....
.....	..... 60, 183, 320, 526
26, 62, 68, 88	\l_@@_fontfeat_sc_clist . 64, 201, 460
\g_@@_em_prop ... 60, 70, 77, 78, 94, 102	\l_@@_fontfeat_sl_clist . 62, 191, 321
\l_@@_em_switch_tl .....	\l_@@_fontfeat_up_clist .....
102, 103, 112	..... 58, 172, 207, 318
\l_@@_em_tmp_tl .....	\g_@@_fontid_family_prop 72, 241, 269
94, 96, 113	\l_@@_fontid_tl .....
\l_@@_emdef_int .....	..... 21, 23, 40, 93, 237, 241, 249
34, 61, 77, 78, 79	\l_@@_fontname_bf_tl .....
\l_@@_emshape_query_tl .....	..... 75, 125, 142, 300, 306, 319, 549
.....	\l_@@_fontname_bfit_tl .....
86, 90, 94, 97, 111	..... 76,
\@@_error:n .....	127, 153, 299, 300, 301, 322, 535, 557
1, 10, 451	\l_@@_fontname_bfsl_tl .....
\@@_error:nn .....	..... 129, 157, 314, 323
2, 3, 12, 14, 18,	\l_@@_fontname_it_tl .....
24, 30, 36, 67, 73, 79, 85, 138, 382, 422	..... 74, 112, 126, 299, 311, 320, 527
\g_@@_euenc_bool 9, 10, 18, 23, 36, 39, 58	\l_@@_fontname_sc_tl 130, 167, 464, 476
\l_@@_ext_filename_tl 85, 86, 89, 90, 90	\l_@@_fontname_sl_tl 117, 128, 314, 321
\l_@@_extension_tl .....	\l_@@_fontname_tl .. 94, 152, 154, 158
.....	\l_@@_fontname_up_tl .....
39, 45, 53, 72, 90, 91, 156	..... 9, 40,
\l_@@_extensions_clist 47, 48, 61, 254	43, 102, 124, 126, 135, 137, 138, 140, 142
\l_@@_external_bool ... 22, 28, 40, 383	\@@_fontname_wrap:n .....
\@@_extract_all_features: .....	..... 44, 147, 148, 164, 168
94	\l_@@_fontopts_clist .....
\@@_extract_all_features:n ... 20, 94	.....
\l_@@_fake_embolden_tl .....	.. 48, 102, 103, 113, 410, 418, 419, 420
.....	
123, 530, 533, 547	
\l_@@_fake_slant_tl . 122, 525, 552, 555	
\l_@@_family_fontopts_clist ....	
.....	
49, 105, 106, 112	

\g_@@_fontopts_prop .....	\@@_main_IfontFeatureActiveTF:nnn
65, 87, 102, 105, 132, 135, 139, 140, 418	..... 110, 257
\@@_get_features:Nn .....	\@@_main_addfontfeatures:n 66, 70, 144
57, 200	\@@_main_aliasfontfeature:nn . 90, 201
\@@_get_features:n ..	\@@_main_aliasfontfeatureoption:nnn
28, 200, 273, 489	..... 94, 220
\l_@@_graphite_bool .	\@@_main_fontspec:nn .....
10, 217, 360, 373	1, 3
\c_@@_hexcol_tl .....	\@@_main_newAATfeature:nnnn ..
139, 230, 659	78, 175
\l_@@_hexcol_tl .....	\@@_main_newfontface:nnn . . . .
..... 95, 229, 231, 391, 395, 408, 659	..... 55, 112
\@@_if_autofont:nn .....	\@@_main_newfontfamily:nnnN . . . .
397	..... 43, 47, 51, 99
\@@_if_autofont:nnTF .....	\@@_main_newfontfeature:nn . . .
389	74, 168
\@@_if_detect_external:n .....	\@@_main_newopentypefeature:nnn
58	..... 82, 86, 185
\@@_if_detect_external:nnTF 12, 58, 168	\@@_main_setboldmathrm:nn . . .
\@@_if_font_feature:n .....	27, 67
263	\@@_main_setmainfont:nn .....
\@@_if_font_feature:nnTF .....	7, 8, 39
261	\@@_main_setmathrm:nn .....
\@@_if_merge_shape:n .....	23, 61
25	\@@_main_setmathsf:nn .....
\@@_if_merge_shape:nnTF .....	31, 73
21, 182	\@@_main_setmathtt:nn .....
\@@_info:n .....	35, 79
7, 127, 187, 289	\@@_main_setmonofont:nn .....
\@@_info:nn .....	18, 43
8, 391	\@@_main_setsansfont:nn .....
\@@_info:nnn .....	13, 25
9, 278	\@@_make_AAT_feature:nn . 7, 10, 76, 87
\@@_init: .....	\@@_make_AAT_feature_string:Nnn . 22
6, 167, 268, 633	\@@_make_AAT_feature_string:NnnTF
\@@_init_fontface: .....	..... 12, 15, 22, 610
203, 654	\@@_make_OT_feature:nnn .....
\@@_init_ttc:n .....	32, 50, 75, 201, 205, 217, 228, 249, 258
17, 70	\@@_make_font_shapes:Nnnnn . . 330, 405
\@@_int_mult_truncate:Nn . . . .	\@@_make_ot_smallcaps:TF .....
74, 420	598
\@@_iv_str_to_num:Nn .....	\@@_make_smallcaps:TF . . 466, 598, 604
..... 67, 68, 85, 113, 117, 160, 664	\l_@@_mapping_tl .....
\@@_iv_str_to_num:w . . . .	..... 22, 23, 26, 135, 226, 227, 435, 439
671, 672, 674	\g_@@_math_bool .....
\@@_keys_define_code:nnn .....	5,
. 7, 13, 16, 20, 24, 36, 37, 46, 78, 83,	6, 17, 97, 99, 100, 101, 102, 103, 104,
88, 95, 100, 104, 115, 119, 124, 151,	105, 106, 109, 111, 112, 115, 117,
155, 159, 170, 174, 181, 185, 189,	118, 119, 120, 121, 122, 123, 124, 125
193, 197, 204, 209, 215, 219, 223,	\g_@@_math_euler_bool .....
227, 231, 235, 239, 243, 262, 308,	9, 13, 34
314, 334, 355, 359, 363, 387, 417,	\g_@@_math_lucida_bool .....
433, 437, 443, 454, 458, 462, 563, 567	..... 13, 14, 14, 15, 40
\l_@@_keys_leftover_clist .....	\g_@@_mathrm_tl .....
..... 52, 124, 127, 128, 129,	63,
205, 206, 210, 212, 216, 218, 222, 223	64, 73, 74, 76, 77, 80, 83, 96, 114, 118
\@@_keys_set_known:nnN .....	\g_@@_mathsf_tl .....
. . 67, 122, 127, 129, 204, 206, 344, 408	..... 75, 76, 78, 90, 97, 116, 119
\l_@@_lang_name_tl .....	\g_@@_mathtt_tl .....
86,	..... 79, 81, 82, 91, 98, 117, 120
133, 136, 179, 181, 184, 191, 193, 196	\l_@@_mm_bool . . . .
\l_@@_language_int .....	9, 359, 367, 470, 475
30,	\@@_msg_new:nnn .....
45, 68, 158, 165, 291, 305, 317, 326, 333	13,
\l_@@_leftover_clist . . . .	18, 23, 44, 84, 90, 94, 104, 108, 113,
51, 408, 410	117, 122, 127, 132, 138, 142, 149,
\@@_load_external_fontoptions:Nn	153, 157, 161, 166, 170, 175, 180,
..... 18, 79, 417	184, 192, 196, 200, 204, 208, 213, 218
\@@_load_font: .....	
26, 132	
\@@_load_fontname:n . 409, 414, 455, 476	
\@@_main_DeclareFontExtensions:n	
..... 106, 252	

\@@_msg_new:nnnn	15, 28, 37, 48, 58, 66, 74	\@@_primitive_font_set:Nnn	.....
\l_@@_never_check_bool	.....	.....	<u>1</u> , 25, 136, 399, 400, 421
.....	23, 81, 113, 149, 270	\@@_primitive_font_set_hyphenchar:Nn	.....
\g_@@_nfss_enc_tl	4, 15, 21, 33, 39, 51,	.....	38, 368, 380
.....	57, 84, 107, 237, 276, 501, 508, 537, 645	\l_@@_proceed_bool	.....
\l_@@_nfss_fam_tl	..	27, 54, 63, 67	
98, 239, 241, 254		\l_@@_punctspace_adjust_tl	.....
\g_@@_nfss_family_tl	.....	.....	136, 141, 340, 345, 350, 662
...	41, 85, 161, 186, 243, 254, 255,	\g_@@_rawfeatures_sclist	....
.....	268, 269, 276, 282, 283, 284, 285,	.....	56,
.....	290, 291, 292, 293, 501, 508, 537, 538	142, 276, 288, 490, 496, 621, 630, 656	
\l_@@_nfss_prop	.....	\@@_remove_clashing_featstr:n	..
66, 145, 178		.....	23, 69, 624
\l_@@_nfss_sc_tl	.....	\l_@@_rmfamily_family_tl	9, 10, 16, 144
.....	96, 433, 481, 506, 509, 568	\@@_sanitise_fontname:Nn	.....
\l_@@_nfss_tl	... 97, 432, 456, 502, 556	.....	8, 9, 10, 44, 74, 75, 76, 84, 128
\l_@@_nfssfont_prop	.....	\@@_save_family:nn	.....
67, 325, 348		33, 272	
\l_@@_nobf_bool	.....	\@@_save_family_needed:n	.....
.....	2, 26, 128, 131, 297, 304, 550	233	
\l_@@_noit_bool	.....	\@@_save_family_needed:nTF	... 31, 233
.....	3, 27, 108, 111, 297, 309, 528	\@@_save_fontid_family:nn	249, 259, 271
\l_@@_nosc_bool	4, 163, 166, 462, 473, 479	\@@_save_fontinfo:n	.....
\c_@@_opacity_tl	140, 230, 409, 421, 658	274, 280	
\l_@@_opacity_tl	.....	\l_@@_saved_fontname_tl	102, 434, 447
..	99, 229, 231, 409, 414, 421, 426, 658	\l_@@_scale_tl	.....
\l_@@_optical_size_tl	.....	....	103, 198, 269, 270, 283, 494, 657
.....	100, 159, 467, 484, 640	\l_@@_script_exist_bool	.....
\l_@@_options_tl	... 101, 151, 154, 158	.....	25, 265, 272, 278
\l_@@_ot_bool	.....	\l_@@_script_int	.....
8, 28, 39,		29, 42, 67,	
65, 80, 93, 110, 125, 140, 208, 269,		96, 113, 119, 124, 157, 165, 271, 289, 290	
358, 370, 378, 465, 475, 578, 606, 636		\l_@@_script_name_tl	.....
\@@_ot_compat:nn	... 344, 348, 349,	..	81, 131, 136, 146, 174, 178, 183, 195
.....	350, 351, 352, 353, 354, 355, 356,	\@@_select_font_family:nn	.....
.....	357, 358, 359, 360, 361, 362, 363, 364	.....	<u>1</u> , 43, 153, 154, 157, 160
\g_@@_pkg_euler_loaded_bool	2, 3, 8, 15	\@@_set_autofont:Nnn	.....
\c_@@_postadjust_tl	.....	....	299, 300, 301, 306, 311, 314, 381
141, 660		\@@_set_default_features:nn	.. 60, 116
\l_@@_postadjust_tl	.....	\@@_set_faces:	.....
138, 357,		277, 316	
367, 379, 502, 509, 538, 569, 571, 660		\@@_set_faces_aux:nnnnn	.... 325, 327
\l_@@_pre_feat_sclist	143, 288, 496, 575	\@@_set_font_default_features:nnn	.....
\@@_preparse_features:	.....	.....	61, 121
25, 118		\@@_set_font_dimen:NnN	.. 280, 281, 292
\l_@@_prev_unicode_name_tl	... 57, 62	\@@_set_font_type:N	.....
\l_@@_primitive_font	.....	9,	
25, 26		27, 38, 64, 79, 92, 109, 124, 139, 139, 352	
\@@_primitive_font_glyph_if_-		\@@_set_scriptlang:	.....
exist:Nn	.....	27, 170	
30		\@@_setboldmathrm_hook:nn	... 71, 91
\@@_primitive_font_glyph_if_-		\@@_setmainfont_hook:nn	.....
exist:NnTF	.....	22, 85	
16, 30, 377		\@@_setmathrm_hook:nn	.....
\@@_primitive_font_gset:Nnn	..	65, 88	
<u>1</u> , 141		\@@_setmathsf_hook:nn	.....
\@@_primitive_font_if_exist:nTF	.....	77, 89	
.....	21, 169	\@@_setmathtt_hook:nn	.....
\@@_primitive_font_if_null:NTF	..	83, 90	
.....	13, 26, 138, 422	\@@_setmonofont_hook:nn	.....
\@@_primitive_font_if_null_p:N	..	58, 87	
<u>13</u>		\@@_setsansfont_hook:nn	.....
		40, 86	
		\@@_setup_nfss:Nnnn	.... 456, 481, 485
		\@@_setup_single_size:nn	... 436, 444

\l_@@_sffamily_family_tl	27, 28, 34, 145
\@@_shape_merge:nn	10, 11, 12, 13, 14, 15, 16, 17, 18, 22, 28, 33, 184, 515, 516
\g_@@_single_feat_tl	58, 73, 86, 265, 273, 277, 279, 281, 306, 318, 328, 335, 619
\l_@@_size_tl	104, 229, 446, 451, 452, 487, 494
\l_@@_sizedfont_tl	105, 233, 447, 455, 457
\l_@@_sizefeat_clist	45, 46, 206, 211, 343, 349
\l_@@_sizing_leftover_clist	53, 450, 456, 482
\l_@@_smcp_shape_tl	184, 187, 190, 193
\@@_strip_leading_sign:Nw	666, 668
\@@_strip_plus_minus:n	194, 196
\@@_strip_plus_minus_aux:Nq	196, 197
\l_@@_strnum_int	31, 85, 91, 117, 124, 160, 166, 271, 289, 305, 326, 333
\l_@@_strong_int	35, 134, 137, 138, 141, 143, 146, 157, 160
\g_@@_strong_prop	71, 116, 124, 125, 135, 143
\l_@@_strong_switch_tl	143, 144
\l_@@_strong_tmp_tl	135, 137
\l_@@_strongdef_int	36, 117, 124, 125, 126
\@@_swap_plus_minus:n	70, 76
\@@_swap_plus_minus_aux:Nq	76, 77
\l_@@_tfm_bool	6, 356, 362
\l_@@_this_feat_clist	57, 247, 255, 260
\l_@@_this_font_tl	106, 212, 213, 217, 245, 254, 260, 340, 346, 349
\l_@@_tmp_int	32, 419, 420, 428, 429
\l_@@_tmp_tl	41, 42, 44, 45, 95, 96, 104, 105, 106, 107, 127, 127, 128, 128, 132, 135, 137, 138, 139, 140, 141, 142, 143, 150, 180, 181, 185, 241, 243, 247, 248, 249, 261, 263, 264, 266, 267, 268, 344
\l_@@_tmpa_bool	19, 63, 66, 68
\l_@@_tmpa_dim	39, 280, 285
\l_@@_tmpa_fp	37
\l_@@_tmpa_tl	24, 25, 48, 108
\l_@@_tmpb_dim	40, 281, 286
\l_@@_tmpb_fp	38
\l_@@_tmpb_tl	30, 35, 38, 42, 45, 48, 109, 132, 133, 134, 135
\l_@@_tmpc_dim	41
\@@_trace:n	10
\l_@@_ttc_index_tl	92, 93, 97, 98, 110, 157, 641
\l_@@_ttfamily_family_tl	45, 46, 52, 146
\@@_update_featstr:n	17, 72, 172, 227, 231, 361, 456, 460, 472, 491, 499, 508, 565, 569, 614
\@@_warning:n	3, 4, 13, 29, 35, 91, 447, 451, 478, 516
\@@_warning:nn	5, 19, 61, 62, 66, 164, 218, 250, 282, 287, 292, 309, 338, 371, 401, 412, 424
\@@_warning:nnn	6, 34, 181, 191
\l_@@_wordspace_adjust_tl	137, 141, 318, 326, 661
\\	17, 25, 33, 33, 34, 37, 38, 39, 97, 98, 99, 110, 146, 163, 172, 177, 187, 188, 189, 210, 215, 221, 222, 223, 545, 557, 571
\_	34
A	
\acute	18
\addfontfeature	30, 46, 68, 96, 96, 100
\addfontfeatures	64, 76, 144
\advance	73
\aliasfontfeature	35, 88, 90, 201, 208, 221, 416
\aliasfontfeatureoption	55, 56, 57, 92, 220, 346
\AtBeginDocument	43, 53, 111, 123, 131
\author	36
B	
\bar	22
\bfdefault	48, 72, 77, 80, 83, 87, 90, 91, 136, 137, 140, 319, 322, 323, 549, 552, 553, 561, 563, 565
\bfseries	161
\bgroup	37
\boldmath	27
bool commands:	
\bool_gset_false:N	3, 6, 8, 10, 68, 97, 99, 100, 101, 102, 103, 104, 105, 106, 111, 112, 117, 118, 119, 120, 121, 122, 123, 124
\bool_gset_true:N	2, 5, 7, 9, 9, 13, 14, 15, 36, 62
\bool_if:NTF	1, 8, 10, 11, 17, 23, 23, 28, 29, 34, 35, 39, 39, 40, 40, 58, 58, 65, 66,

67, 68, 72, 78, 80, 81, 81, 84, 88, 93,  
97, 98, 107, 110, 113, 125, 125, 131,  
140, 143, 149, 172, 173, 189, 199,  
208, 217, 249, 278, 304, 309, 310,  
383, 400, 411, 423, 462, 465, 470,  
477, 479, 523, 545, 578, 606, 609, 617  
\bool\_if:nTF ... 217, 297, 475, 531, 670  
\bool\_lazy\_and:nnTF ..... 27  
\bool\_new:N ..... 1, 2, 3, 4,  
5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17,  
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28  
\bool\_set\_false:N .. 18, 22, 29, 57,  
61, 63, 63, 66, 88, 109, 111, 115, 121,  
131, 162, 166, 204, 222, 265, 271,  
356, 357, 358, 359, 360, 528, 550, 636  
\bool\_set\_true:N . 12, 26, 27, 28, 54,  
55, 59, 65, 66, 92, 108, 125, 128, 153,  
163, 167, 211, 229, 269, 270, 272,  
362, 364, 367, 370, 373, 378, 473, 637  
\bool\_until\_do:nn ..... 89, 122, 163  
\breve ..... 23

## C

\char ..... 5, 6, 17  
char commands:  
  \char\_set\_catcode\_active:n ... 28, 30  
  \char\_set\_catcode\_ignore:n ..... 226  
  \char\_set\_catcode\_space:n ..... 17  
\check ..... 24  
clist commands:  
  \clist\_clear:N ..... 103, 106, 272, 419  
  \clist\_count:N ..... 248  
  \clist\_count:n ..... 100  
  \clist\_gput\_right:Nn ..... 118  
  \clist\_gset:Nn ..... 1, 118  
  \clist\_map\_break: ..... 54, 66, 274  
  \clist\_map\_inline:Nn . 48, 61, 206, 224  
  \clist\_map\_inline:nn ..... 74, 74,  
  86, 121, 124, 204, 215, 235, 266, 436, 627  
  \clist\_new:N .....  
  43, 44, 45, 47, 48, 49, 50, 51, 52, 53,  
  54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64  
  \clist\_put\_left:Nn ..... 469  
  \clist\_put\_right:Nn .....  
  .... 207, 526, 532, 534, 548, 554, 556  
  \clist\_set:Nn .....  
  ... 46, 99, 109, 172, 176, 183, 187,  
  191, 195, 201, 206, 211, 246, 254, 343  
  \clist\_set\_eq:NN ..... 247, 460  
\colon ..... 30, 31, 112  
\convertcolorspec ..... 391

## cs commands:

\cs:w ..... 127  
\cs\_end: ..... 127  
\cs\_generate\_variant:Nn .....  
  ..... 11, 12, 73, 75,  
  78, 79, 80, 81, 82, 83, 84, 85, 86, 87,  
  88, 89, 90, 91, 92, 156, 271, 443, 528, 684  
\cs\_gset:Npn ..... 29  
\cs\_if\_eq:NNTF ..... 103, 136, 179  
\cs\_if\_exist:NTF ..... 3, 27, 190, 389  
\cs\_if\_exist\_p:N ..... 30  
\cs\_new:Nn ..... 1, 1, 1, 3, 4, 5,  
  7, 7, 10, 10, 11, 13, 14, 15, 15, 19, 20,  
  25, 37, 38, 38, 39, 43, 44, 45, 50, 61,  
  67, 67, 70, 73, 74, 76, 79, 79, 93, 94,  
  99, 112, 116, 118, 121, 132, 144, 145,  
  150, 152, 157, 159, 160, 162, 166,  
  168, 170, 175, 185, 196, 200, 201,  
  220, 252, 257, 259, 262, 272, 272,  
  280, 292, 295, 299, 316, 327, 332,  
  338, 344, 352, 381, 405, 414, 429,  
  444, 485, 499, 504, 513, 519, 529,  
  541, 598, 599, 604, 614, 624, 654, 664  
\cs\_new:Npn ..... 1, 2,  
  3, 4, 5, 6, 7, 8, 9, 10, 65, 66, 77, 197, 225  
\cs\_new\_protected:Nn ..... 1  
\cs\_new\_protected:Npn ..... 58, 114  
\cs\_set:Npn ..... 1, 5, 9, 50, 54, 60,  
  61, 66, 69, 71, 80, 87, 111, 112, 113,  
  152, 153, 168, 314, 406, 633, 668, 674  
\cs\_set\_eq:NN ..... 31, 43, 60, 63,  
  84, 85, 86, 87, 88, 89, 90, 91, 91, 164, 173  
\cs\_to\_str:N 101, 105, 106, 127, 138, 181  
\cs\_undefine:N ..... 86, 255, 524  
\cyrillicencoding ..... 51, 55, 79

## D

\date ..... 56  
\ddot ..... 20  
\DeclareDocumentCommand .....  
  . 9, 15, 21, 27, 33, 39, 51, 64, 70, 76, 82  
\DeclareErrorFont ..... 42  
\DeclareFontEncoding ..... 30, 41  
\DeclareFontExtensions ..... 104  
\DeclareFontFamily ..... 44, 276  
\DeclareFontShape .....  
  ..... 46, 48, 50, 51, 52, 52, 54, 526  
\DeclareFontSubstitution ..... 31, 43  
\DeclareMathAccent .....  
  ... 18, 19, 20, 21, 22, 23, 24, 25, 26, 27  
\DeclareMathDelimiter ... 65, 66, 67, 68, 69



\fontdimen8 .....	76	\fontspec_if_opentype: .....	23
\fontencoding .....	4, 15, 33, 51, 107, 276	\fontspec_if_opentype:TF .....	23
\fontfamily .....	16, 28, 34, 52, 106, 161, 277	\fontspec_if_script:n .....	75
\fontname .....	38, 161, 177, 401	\fontspec_if_script:nTF .....	75
\fontshape .....	7, 22, 23	\fontspec_if_small_caps: .....	180
\fontspec .....	1, 24, 30, 40, 123	\fontspec_if_small_caps:TF .....	180
fontspec commands:		\l_fontspec_lang_tl .....	48, 134, 186, 293, 304, 316, 327, 334, 583, 594
\fontspec_calc_scale:n .....	75	\fontspec_maybe_setup_maths: ....	93
\fontspec_complete_fontname:Nn .		\fontspec_merge_shape:n .....	
.....	102, 112, 117, 132, 153, 157, 167, 233, 329, 332	.....	19, 41, 46, 51, 56
\g_fontspec_default_fontopts_tl .	30	\l_fontspec_mode_tl .....	69, 73, 81, 590, 647
\g_fontspec_encoding_tl .....		\fontspec_new_lang:nn .....	102, 299
.....	23, 41, 42, 44, 48, 49, 51, 52, 55, 56, 73, 74, 75, 76, 77, 78, 79, 80, 83, 86, 87, 88, 90, 91, 645	\fontspec_new_script:nn .....	98, 262
\l_fontspec_family_tl .....		\fontspec_parse_colour:niii .	398, 406
.....	40, 41, 74, 154, 162	\fontspec_parse_cv:w .....	225, 237
\l_fontspec_feature_string_tl .....	17, 48	\_fontspec_parse_wordspace:w	311, 314
\fontspec_font_if_exist:n .....	164	\fontspec_patch_fancyvrb: ...	47, 80
\fontspec_font_if_exist:nTF ....	173	\fontspec_patch_listings: ...	48, 87
\l_fontspec_fontname_tl .....		\fontspec_patch_moreverb: ...	46, 66
.....	8, 18, 21, 40, 43, 46, 77, 102, 110, 115, 120, 125, 126, 130, 135, 140, 145, 198, 206, 287, 301, 306, 311, 318, 417, 418, 421, 426, 431, 434, 487, 495, 527, 535, 549, 557	\fontspec_patch_verbatim: ...	45, 50
\l_fontspec_hyphenchar_tl .....	374, 375, 377, 380	\fontspec_print_visible_spaces:	
\fontspec_if_aat_feature:nn .....	5	.....	27, 42, 56, 62, 76
\fontspec_if_aat_feature:nnTF ....	5	\l_fontspec_renderer_tl .....	
\fontspec_if_current_feature:n .	174	. ....	50, 54, 58, 76, 158, 365, 371, 374, 642
\fontspec_if_current_feature:nTF		\l_fontspec_script_tl .....	
.....	174, 281	.....	47, 97, 112, 132, 141, 186, 270, 288, 292, 580, 582, 591, 593
\fontspec_if_current_language:n	135	\fontspec_select:nn .....	43
\fontspec_if_current_language:nTF		\fontspec_set_em_level:n .....	159
.....	135	\fontspec_set_family:Nnn .	3, 9, 27, 45, 63, 64, 69, 70, 75, 76, 81, 82, 101, 150
\fontspec_if_current_script:n ..	120	\fontspec_set_fontface:NNnn .....	157
\fontspec_if_current_script:nTF	120	\fontspec_set_strong_level:n ...	160
\fontspec_if_feature:n .....	34	\fontspec_setup_maths: .....	1, 128
\fontspec_if_feature:nnn .....	60	\fontspec_tmp: .....	60, 63
\fontspec_if_feature:nnnTF .....	60	\fontspec_visible_space: .....	14, 31, 84, 91
\fontspec_if_feature:nTF .....	34	\fontspec_visible_space_fallback:	
\fontspec_if_fontspec_font: .....	1	.....	18, 20
\fontspec_if_fontspec_font:TF .	1, 7, 25, 36, 62, 77, 90, 107, 122, 137, 147	fontspec internal commands:	
\fontspec_if_language:n .....	88	\l_fontspec_check_bool .....	
\fontspec_if_language:nn .....	105	.....	88, 92, 98, 107, 121, 125, 131, 143
\fontspec_if_language:nnTF .....	105	\_fontspec_update_featstr:n ....	97
\fontspec_if_language:nTF .....	88	\FONTSPECCTX .....	2
		\FontspecSetCheckBoolFalse .....	65
		\FontspecSetCheckBoolTrue .....	65
		fp commands:	
		\fp_eval:n .....	285
		\fp_new:N .....	37, 38

<b>G</b>		$\backslash$ l_tmpb_int	
$\backslash$ g	113		86, 89, 93, 118, 122, 126, 154, 163, 168
$\backslash$ Gamma	52	$\backslash$ c_zero	366
$\backslash$ gdef	2	$\backslash$ itdefault	2, 11, 13, 15, 41, 76, 83, 88, 320, 322, 533, 534, 538, 550, 552
$\backslash$ GetFileInfo	55	$\backslash$ itscdefault	2, 7, 11, 13, 15, 16, 17, 562, 563
$\backslash$ global	7, 73, 161	$\backslash$ itshape	38, 66, 112
$\backslash$ grave	19	<b>K</b>	
group commands:		keys commands:	
$\backslash$ group_begin:	4, 23, 27, 28, 56, 72, 119, 149, 166, 266, 274, 407, 522	$\backslash$ l_keys_choice_int	52, 56, 71
$\backslash$ group_end:	27, 28, 33, 33, 39, 81, 128, 160, 170, 171, 274, 290, 411, 525	$\backslash$ keys_define:nn	
<b>H</b>			2, 3, 5, 9, 20, 20, 22, 27, 47, 69, 81, 92, 170, 197, 210, 212, 230, 231, 237, 244, 244, 253, 261, 264, 298, 301, 320, 424, 487, 495, 504, 514, 519, 541
$\backslash$ hat	25, 111	$\backslash$ keys_if_choice_exist:nnnTF	180, 190
$\backslash$ hbox	36	$\backslash$ keys_if_exist:nnTF	
$\backslash$ hyphenchar	6, 9		177, 187, 208, 226, 234, 241
<b>I</b>		$\backslash$ l_keys_key_tl	119, 124, 129, 134
$\backslash$ IfBooleanTF	118, 130	$\backslash$ keys_set:nn	9, 13, 32, 42, 79, 80, 85, 183, 184, 195, 196, 212, 213, 218, 223, 231, 238, 245, 310, 339, 449
$\backslash$ ifcase	361	$\backslash$ keys_set_groups:nnn	420
$\backslash$ IfFontExistsTF	173	$\backslash$ keys_set_known:nn	15
$\backslash$ IfFontFeatureActiveTF	108, 257	$\backslash$ keys_set_known:nnN	70, 80, 150, 449
$\backslash$ ifmmode	36	$\backslash$ l_keys_value_tl	119, 124, 129, 134
$\backslash$ IfNoValueTF	59	<b>L</b>	
$\backslash$ ifnum	6, 91, 124, 165, 366	$\backslash$ l	30, 49, 52, 61, 62, 62, 67
$\backslash$ ifx	15, 29, 30, 97, 109, 115, 680, 681	$\backslash$ Lambda	55
$\backslash$ ignorespaces	4, 9, 14, 19, 62, 166	$\backslash$ latinencoding	52, 56, 80
$\backslash$ InputIfFileExists	3	$\backslash$ leavevmode	36
int commands:		$\backslash$ let	38
$\backslash$ int_case:nn	56, 71	$\backslash$ liningnums	94
$\backslash$ int_case:nnTF	300	listingcont*(environment)	66
$\backslash$ int_compare:nTF		$\backslash$ LuaLaTeX	34
	28, 52, 100, 248, 394, 397, 428	luatex commands:	
$\backslash$ int_compare_p:nNn	89, 122, 163	$\backslash$ luatex_postexhyphenchar:D	651
$\backslash$ int_eval:n	264	$\backslash$ luatex_posthyphenchar:D	649
$\backslash$ int_if_even:nTF	33	$\backslash$ luatex_preexhyphenchar:D	650
$\backslash$ int_incr:N	79, 95, 100, 126, 128, 141, 170	$\backslash$ luatex_prehyphenchar:D	648
$\backslash$ int_new:N	29, 30, 31, 32, 33, 34, 35, 36	<b>M</b>	
$\backslash$ int_set:Nn	42, 45, 76, 78, 86, 93, 96, 96, 118, 126, 137, 154, 159, 160, 168, 271, 289, 305, 326, 333, 419, 648, 676	$\backslash$ mathalpha	
$\backslash$ int_set_eq:NN	11		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
$\backslash$ int_to_hex:n	429	$\backslash$ mathbf	56, 77, 87, 113
$\backslash$ int_use:N		$\backslash$ mathbin	63
	78, 93, 97, 102, 125, 134, 138, 143	$\backslash$ mathchardef	29
$\backslash$ int_zero:N	61, 87, 105, 117, 120, 146, 156, 157, 161, 317, 649, 650, 651	$\backslash$ mathclose	36, 39, 66, 68
$\backslash$ c_one	11, 73		
$\backslash$ l_tmpa_int	87, 89, 91, 93, 95, 120, 122, 124, 126, 128, 161, 163, 166, 168, 170		



123, 124, 125, 126, 127, 128, 129,  
 130, 131, 132, 133, 134, 135, 136,  
 137, 138, 139, 140, 141, 142, 143, 144  
 \newICUfeature ..... 84  
 \newopentypefeature ..... 80, 185  
 \normalfont ..... 23, 41, 59, 73, 120, 155  
 \normalsize ..... 50, 523  
 \null ..... 36  
 \nullfont ..... 15  
 \numexpr ..... 40  
  
 O  
 \oldstylenums ..... 4, 94, 124  
 \Omega ..... 62  
 \or ..... 363, 369, 372  
  
 P  
 \par ..... 74  
 Path ..... 24  
 \Phi ..... 60  
 \Pi ..... 57  
 prg commands:  
   \prg\_new\_conditional:Nnn .....  
     ..... 1, 5, 22, 23, 25, 30,  
     34, 58, 60, 75, 79, 88, 105, 111, 120,  
     135, 147, 164, 174, 180, 233, 263, 397  
   \prg\_return\_false: .....  
     ..... 3, 13, 16, 18, 20, 26,  
     27, 28, 31, 35, 36, 46, 50, 53, 57, 68,  
     69, 71, 73, 82, 84, 86, 98, 99, 101, 103,  
     107, 114, 116, 118, 129, 131, 131,  
     133, 143, 144, 146, 148, 171, 173,  
     178, 189, 195, 198, 244, 279, 282, 402  
   \prg\_return\_true: .....  
     ..... 3, 13, 16, 28, 28, 33, 36,  
     49, 50, 68, 69, 82, 82, 98, 99, 107, 114,  
     114, 129, 131, 143, 144, 150, 170,  
     173, 178, 189, 196, 250, 256, 282, 403  
   \prg\_set\_conditional:Nnn ..... 13, 21  
 \ProcessOptions ..... 22  
 prop commands:  
   \prop\_gclear:N ..... 60, 116  
   \prop\_get:NnN .....  
     41, 44, 47, 48, 95, 97, 127, 142, 151, 152  
   \prop\_get:NnNTF ... 85, 86, 94, 102,  
     102, 105, 132, 135, 143, 241, 261, 418  
   \prop\_gput:Nnn ..... 11, 78,  
     84, 125, 135, 140, 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, 267, 268, 269, 269, 270, 271,  
 272, 273, 274, 275, 276, 277, 278,  
 279, 280, 281, 282, 283, 283, 284,  
 284, 285, 285, 286, 287, 288, 289,  
 290, 290, 291, 291, 292, 292, 293,  
 293, 294, 295, 296, 297, 298, 299,  
 300, 301, 302, 303, 304, 305, 306,  
 307, 308, 309, 310, 311, 312, 313, 314  
 \prop\_gput\_if\_new:Nnn .... 77, 83, 124  
 \prop\_gremove:Nn ..... 139  
 \prop\_if\_in:NnTF ..... 9, 87  
 \prop\_map\_inline:Nn ..... 325  
 \prop\_new:N .....  
   .. 65, 66, 67, 68, 69, 70, 71, 72, 73, 282  
 \prop\_put:Nnn .... 81, 82, 145, 178, 348  
 \providecommand ..... 1, 2, 2, 3, 3, 4, 5, 6  
 \ProvidesExplFile ..... 49  
 \ProvidesExplPackage ..... 44, 45, 46  
 \Psi ..... 61  
  
 Q  
 \qqquad ..... 56  
 quark commands:  
   \q\_nil ..... 76, 77, 196,  
     197, 226, 238, 666, 668, 671, 672, 674  
   \q\_stop ..... 311, 314  
  
 R  
 \relax ..... 5, 6, 29, 36, 40, 45, 50, 55  
 \RenewDocumentCommand ..... 47, 94  
 \renewfontfamily ..... 45  
 \RequirePackage ..... 5, 43, 48, 48, 62  
 \RequirePackageWithOptions ..... 7, 12  
 \rmdefault . 10, 20, 28, 45, 96, 113, 118, 277  
 \rmfamily ..... 13, 76, 299  
  
 S  
 scan commands:  
   \scan\_stop: ..... 3, 7, 17, 32, 40, 161  
 \scdefault ..... 2, 3, 11, 12,  
   13, 14, 17, 18, 51, 515, 516, 517, 560, 561  
 \scshape ..... 38  
 \select ..... 19  
 \selectfont .....  
   .. 5, 7, 17, 22, 23, 35, 53, 108, 161, 278

# seq commands:

<code>\seq_if_empty:NTF</code>	134
<code>\seq_new:N</code>	42
<code>\seq_put_right:Nn</code>	122, 137
<code>\setboldmathrm</code>	25, 27, 67, 93, 113
<code>\setfontfamily</code>	49
<code>\setmainfont</code>	6, 7, 24, 27, 111
<code>\SetMathAlphabet</code>	...
	76, 77, 78, 79, 83, 86, 87, 88, 90, 91
<code>\setmathrm</code>	21, 61, 92, 113
<code>\setmathsf</code>	29, 73, 94
<code>\setmathtt</code>	33, 79, 95
<code>\setmonofont</code>	16, 43
<code>\setromanfont</code>	37
<code>\setsansfont</code>	11, 25
<code>\SetSymbolFont</code>	17, 74, 80
<code>\settoheight</code>	297
<code>\sfdefault</code>	28, 38, 46, 97, 119
<code>\sffamily</code>	31
<code>\Sigma</code>	58
<code>\sishape</code>	2
<code>\sldefault</code>	3, 12,
	14, 16, 46, 321, 323, 534, 537, 551, 553
<code>\slscdefault</code>	3, 12, 14, 15, 16, 18, 564, 565
<code>\slshape</code>	38, 64
<code>\space</code>	30, 35, 40, 198

# str commands:

<code>\c_backslash_str</code>	87
<code>\c_colon_str</code>	226, 238
<code>\str_case:nn</code>	78, 546, 558
<code>\str_case:nnTF</code>	199, 264
<code>\str_case_x:nnTF</code>	336
<code>\str_if_eq:nnTF</code>	...
	87, 128, 143, 280, 299, 365, 445
<code>\str_if_eq_p:nn</code>	670
<code>\str_if_eq_x:nnTF</code>	...
	20, 38, 56, 72, 90, 150, 229, 401
<code>\str_if_eq_x_p:nn</code>	533, 534
<code>\str_lower_case:n</code>	72, 90
<code>\string</code>	21, 56, 76, 96
<code>\strong</code>	151
<code>\strongenv</code>	130, 151
<code>\strongfontdeclare</code>	114, 161
<code>\strongreset</code>	121, 147, 151

# sys commands:

<code>\sys_if_engine luatex:TF</code>	3
<code>\sys_if_engine xetex:TF</code>	10

# T

## T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> commands:

<code>\@</code>	30, 57, 61, 62
<code>\@sverb</code>	40, 42
<code>\@filelist</code>	50
<code>\@ifpackageloaded</code>	1, 6, 13, 14, 15, 52,
	68, 82, 89, 95, 99, 100, 101, 102, 103,
	104, 105, 106, 107, 111, 112, 113,
	117, 118, 119, 120, 121, 122, 123, 124
<code>\@ifstar</code>	40
<code>\@makeother</code>	38
<code>\@noligs</code>	39
<code>\@nomath</code>	85, 132
<code>\@onlypreamble</code>	92, 93, 94, 95
<code>\@sverb</code>	42
<code>\@sxverbatim</code>	62
<code>\@tempa</code>	29, 30
<code>\@verb</code>	40
<code>\@verbatim</code>	56, 62, 76
<code>\add@unicode@accent</code>	5, 7, 19
<code>\color@</code>	389
<code>\curr@fontshape</code>	104, 137, 180
<code>\define@ant@mathversions</code>	97
<code>\define@iwona@mathversions</code>	109
<code>\define@kurier@mathversions</code>	115
<code>\f@encoding</code>	32, 190, 193, 194
<code>\f@family</code>	3, 3, 32, 41, 44, 47, 48,
	95, 97, 127, 142, 151, 152, 190, 193, 194
<code>\f@series</code>	...
	23, 32, 124, 135, 138, 190, 193, 194
<code>\f@shape</code>	22, 23, 28, 33, 77, 86, 184
<code>\f@size</code>	104,
	137, 137, 142, 180, 399, 400, 421, 524
<code>\listing@line</code>	73
<code>\lst@visible space</code>	91
<code>\not@math@alphabet</code>	6, 40, 45, 50, 55
<code>\reset@font</code>	153
<code>\thelisting@line</code>	73
<code>\two@digits</code>	217, 229
<code>\verb@eol@error</code>	38
<code>\verbatim@font</code>	39
<code>\verbatim@line</code>	74
<code>\verbatim@processline</code>	71
<code>\verbatim@start</code>	56, 76
<code>\x@defaultthyphenchar</code>	7, 13
<code>\z@</code>	6

## tex commands:

<code>\tex_hyphenchar:D</code>	40
<code>\textsc</code>	37
<code>\textsf</code>	33
<code>\textsi</code>	2

<code>\textvisiblespace</code> .....	24	86, 92, 93, 97, 98, 104, 105, 112, 127,	
<code>\the</code> .....	74	137, 138, 152, 159, 178, 180, 181,	
<code>\Theta</code> .....	54	181, 187, 193, 213, 217, 229, 241,	
<code>\tilde</code> .....	21	247, 263, 266, 269, 270, 270, 288,	
<code>\title</code> .....	32	304, 316, 318, 326, 327, 334, 334,	
tl commands:		340, 340, 345, 350, 365, 371, 374,	
<code>\c_empty_tl</code> .....	671, 672, 680, 681	374, 375, 390, 395, 408, 414, 426,	
<code>\tl_clear:N</code> .....		435, 439, 467, 484, 525, 547, 575, 647	
23, 45, 107, 133, 245, 255, 432, 433,		<code>\tl_set_eq:NN</code> .....	10, 21, 28,
446, 639, 640, 641, 642, 657, 661, 662		39, 41, 46, 49, 51, 52, 55, 56, 57, 57,	
<code>\tl_clear_new:N</code> .....	78, 79, 80	62, 126, 142, 154, 162, 184, 254, 434,	
<code>\tl_const:Nn</code> .....	11,	447, 527, 535, 549, 557, 658, 659, 660	
12, 13, 14, 15, 16, 17, 18, 139, 140, 141		<code>\tl_to_str:N</code> .....	21
<code>\tl_count:n</code> .....	394, 397	<code>\tl_to_str:n</code> .....	87, 177
<code>\tl_gclear:N</code> .....	265, 643, 644, 656	<code>\tl_trim_spaces:n</code> .....	14, 16
<code>\tl_gput_right:Nn</code> .....	543, 621	<code>\tl_use:N</code> .....	22, 33, 516
<code>\tl_gremove_all:Nn</code> .....	630	<code>\tmpa</code> .....	28, 29
<code>\tl_gset:Nn</code> ...	41, 42, 44, 58, 73, 96,	token commands:	
97, 98, 118, 119, 120, 121, 136, 237,		<code>\token_to_str:N</code> .....	87, 389
268, 273, 283, 306, 318, 328, 335, 619		<code>\tracingall</code> .....	211
<code>\tl_gset_eq:NN</code> .....	243, 254, 645	<code>\ttdefault</code> .....	46, 47, 56, 98, 120
<code>\tl_if_empty:NTF</code> .....		<code>\ttfamily</code> .....	49
.. 25, 42, 45, 81, 174, 179, 191, 212,		<code>\typeout</code> .....	3, 5, 23, 31,
226, 239, 279, 346, 365, 371, 374,		37, 52, 60, 69, 71, 83, 93, 96, 97, 120,	
387, 451, 464, 506, 530, 552, 580, 591		123, 134, 134, 135, 138, 140, 146,	
<code>\tl_if_empty:nTF</code> 7, 12, 18, 57, 88, 89,		147, 153, 154, 178, 202, 203, 210,	
90, 106, 126, 138, 161, 316, 342, 385, 569		210, 216, 222, 228, 236, 236, 237,	
<code>\tl_if_eq:NNTF</code> .....	192, 409, 421	243, 250, 253, 259, 260, 276, 277,	
<code>\tl_if_eq:nnTF</code> .....	91, 140	354, 416, 431, 452, 457, 468, 472,	
<code>\tl_if_exist:NTF</code> .....	515	487, 490, 521, 616, 620, 626, 629, 635	
<code>\tl_if_exist_p:N</code> .....	28		
<code>\tl_if_in:NnTF</code> .....	50, 50, 251		
<code>\tl_if_in:nnTF</code> .....	64, 65, 66, 176		
<code>\tl_if_single:NTF</code> .....	126, 373		
<code>\tl_new:N</code> .....			
74, 75, 76, 77, 81, 82, 83, 84, 85, 86,			
87, 88, 89, 90, 91, 92, 93, 94, 95, 96,			
97, 98, 99, 100, 101, 102, 103, 104,			
105, 106, 107, 108, 109, 110, 111,			
112, 113, 114, 115, 116, 117, 121,			
122, 123, 124, 125, 126, 127, 128,			
129, 130, 131, 132, 133, 134, 135,			
136, 137, 138, 142, 143, 144, 145, 146			
<code>\tl_put_left:Nn</code> .....	42		
<code>\tl_put_right:Nn</code> 134, 357, 367, 379, 492			
<code>\tl_remove_all:Nn</code> ...	47, 86, 248, 336		
<code>\tl_remove_once:Nn</code> .....	52		
<code>\tl_replace_all:Nnn</code> .....	90, 92, 335		
<code>\tl_set:Nn</code> .....	21,		
22, 24, 26, 29, 30, 35, 38, 39, 45, 46,			
46, 47, 48, 53, 54, 58, 69, 81, 85, 86,			

## U

<code>\UndeclareAccent</code> .....	64
<code>\UndeclareCommand</code> .....	64
<code>\UndeclareComposite</code> .....	82
<code>\UndeclareSymbol</code> .....	64
<code>\UndeclareTextCommand</code> .....	68, 74, 80
<code>\unexpanded</code> .....	96, 123
<code>\UnicodeEncodingName</code> .....	13,
19, 25, 31, 37, 57, 58, 62, 68, 74, 80, 87	
<code>\UnicodeFontFile</code> ....	1, 47, 49, 51, 53, 55
<code>\UnicodeFontName</code> .....	2
<code>\UnicodeFontTeXLigatures</code> .....	
.....	3, 4, 47, 49, 51, 53, 55
<code>\updefault</code> .	17, 18, 56, 73, 74, 77, 78, 79,
80, 86, 87, 90, 91, 194, 318, 319, 548, 549	
<code>\upshape</code> .....	38, 111, 113
<code>\Upsilon</code> .....	59
<code>\url</code> .....	40
use commands:	
<code>\use:N</code> .....	99, 106

