

# 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://wsprr.io/fontspec/>

2017/11/09 v2.6g

## Contents

<b>I</b>	<b>fontspec.dtx</b>	<b>5</b>
1	Package declaration	5
1.1	Lua header	6
<b>II</b>	<b>fontspec-code-load.dtx</b>	<b>7</b>
1	The fontspec.sty loading file	7
<b>III</b>	<b>fontspec-vars.dtx</b>	<b>8</b>
1	Declaration of variables	8
<b>IV</b>	<b>fontspec-msg.dtx</b>	<b>12</b>
1	Error/warning/info messages	12
1.1	Errors	12
1.2	Warnings	13
1.3	Info messages	15
<b>V</b>	<b>fontspec-opening.dtx</b>	<b>17</b>
1	Opening code	17
1.1	Package options	17
1.2	Encodings	17

1.3	Generic functions	18
1.4	expl3 variants	19
<b>VI</b>	<b>fontspec-fontload.dtx</b>	<b>20</b>
1	expl3 interface for primitive font loading	20
<b>VII</b>	<b>fontspec-interfaces.dtx</b>	<b>22</b>
1	User commands	22
<b>VIII</b>	<b>fontspec-user.dtx</b>	<b>25</b>
1	User command internals	25
1.1	Font selection	25
1.2	Font feature selection	28
1.3	Defining new font features	30
<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	56
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

<b>XIV</b>	<b>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</b>	<b>fontspec-scripts.dtx</b>	<b>93</b>
1	Font script definitions	93
<b>XVI</b>	<b>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
<b>Index</b>		<b>125</b>

# 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 {2017/11/09}{2.6g}{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 = "2.6g",
63 <lua> date = "2017/11/09",
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 $\LaTeX$

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

#### X $\TeX$

```
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
```

## Counters

```
27 \int_new:N \l_@@_script_int
28 \int_new:N \l_@@_language_int
29 \int_new:N \l_@@_strnum_int
30 \int_new:N \l_@@_tmp_int
31 \int_new:N \l_@@_em_int
32 \int_new:N \l_@@_emdef_int
33 \int_new:N \l_@@_strong_int
34 \int_new:N \l_@@_strongdef_int
```

## Floats

```
35 \fp_new:N \l_@@_tmpa_fp
36 \fp_new:N \l_@@_tmpb_fp
```

## Dimensions

```
37 \dim_new:N \l_@@_tmpa_dim
38 \dim_new:N \l_@@_tmpb_dim
39 \dim_new:N \l_@@_tmpc_dim
```

## Sequences

```
40 \seq_new:N \g_@@_bf_series_seq
```

## Comma-lists

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

54 \tl_new:N \l_@@_fontfeat_up_clist
55 \tl_new:N \l_@@_fontfeat_bf_clist
56 \tl_new:N \l_@@_fontfeat_it_clist
57 \tl_new:N \l_@@_fontfeat_bfit_clist
58 \tl_new:N \l_@@_fontfeat_sl_clist
59 \tl_new:N \l_@@_fontfeat_bfsl_clist
60 \tl_new:N \l_@@_fontfeat_sc_clist
```

## Property lists

```
61 \prop_new:N \g_@@_fontopts_prop
62 \prop_new:N \l_@@_nfss_prop
63 \prop_new:N \l_@@_nfssfont_prop
64 \prop_new:N \g_@@_OT_features_prop
65 \prop_new:N \g_@@_all_opentype_feature_names_prop
66 \prop_new:N \g_@@_em_prop
67 \prop_new:N \g_@@_strong_prop
```

## Token lists

```
68 \tl_new:N \l_fontspec_family_tl
69 \tl_new:N \g_fontspec_encoding_tl
70 \tl_new:N \l_fontspec_renderer_tl
71 \tl_new:N \l_fontspec_fontname_tl
72 \tl_new:N \l_fontspec_defined_shapes_tl

73 \tl_clear_new:N \UTFencname
74 \tl_clear_new:N \cyrillicencoding
75 \tl_clear_new:N \latinencoding

76 \tl_new:N \g_@@_single_feat_tl
77 \tl_new:N \l_@@_tmp_tl
78 \tl_new:N \l_@@_size_tl
79 \tl_new:N \l_@@_sizedfont_tl
80 \tl_new:N \l_@@_nfss_tl
81 \tl_new:N \l_@@_nfss_sc_tl
82 \tl_new:N \l_@@_this_font_tl
83 \tl_new:N \l_@@_scale_tl
84 \tl_new:N \l_@@_opacity_tl
85 \tl_new:N \l_@@_hexcol_tl
86 \tl_new:N \l_@@_fontid_tl
87 \tl_new:N \l_@@_extension_tl
88 \tl_new:N \l_@@_ext_filename_tl
89 \tl_new:N \l_@@_font_path_tl
90 \tl_new:N \l_@@_basename_tl
91 \tl_new:N \l_@@_curr_fontname_tl
92 \tl_new:N \l_@@_saved_fontname_tl
93 \tl_new:N \l_@@_optical_size_tl
94 \tl_new:N \l_@@_ttc_index_tl
95 \tl_new:N \l_@@_nfss_enc_tl
96 \tl_new:N \g_@@_curr_series_tl
97 \tl_new:N \l_@@_options_tl
98 \tl_new:N \l_@@_fontname_tl

99 \tl_new:N \g_@@_mathrm_tl
100 \tl_new:N \g_@@_bfmathrm_tl
101 \tl_new:N \g_@@_mathsf_tl
102 \tl_new:N \g_@@_mathtt_tl

103 \tl_new:N \l_@@_family_label_tl
104 \tl_new:N \l_@@_fake_slant_tl
105 \tl_new:N \l_@@_fake_embolden_tl
```

```

106 \tl_new:N \l_@@_fontname_up_tl
107 \tl_new:N \l_@@_fontname_bf_tl
108 \tl_new:N \l_@@_fontname_it_tl
109 \tl_new:N \l_@@_fontname_bfit_tl
110 \tl_new:N \l_@@_fontname_sl_tl
111 \tl_new:N \l_@@_fontname_bfsl_tl
112 \tl_new:N \l_@@_fontname_sc_tl

113 \tl_new:N \l_@@_script_name_tl
114 \tl_new:N \l_fontspeg_script_tl
115 \tl_new:N \l_@@_lang_name_tl
116 \tl_new:N \l_fontspeg_lang_tl

117 \tl_new:N \l_@@_mapping_tl
118 \tl_new:N \g_@@_hexcol_tl
119 \tl_new:N \g_@@_opacity_tl
120 \tl_set:Nn \g_@@_hexcol_tl {000000}
121 \tl_set:Nn \g_@@_opacity_tl {FF~}
122 \tl_new:N \l_@@_punctspace_adjust_tl
123 \tl_new:N \l_@@_wordspace_adjust_tl
124 \tl_new:N \l_@@_postadjust_tl
125 \tl_new:N \g_@@_postadjust_tl
126 \tl_set:Nn \g_@@_postadjust_tl { \l_@@_wordspace_adjust_tl \l_@@_punctspace_adjust_tl }

```

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

```

127 \tl_new:N \l_@@_rawfeatures_sclist
128 \tl_new:N \l_@@_pre_feat_sclist

```

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

```

129 \tl_new:N \g_@@_rmfamily_family
130 \tl_new:N \g_@@_sffamily_family
131 \tl_new:N \g_@@_ttfamily_family

```

## 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:nxx {fontspec} }
4 \cs_new:Npn \@@_warning:n { \msg_warning:nn {fontspec} }
5 \cs_new:Npn \@@_warning:nx { \msg_warning:nxx {fontspec} }
6 \cs_new:Npn \@@_warning:nxx { \msg_warning:nxxx {fontspec} }
7 \cs_new:Npn \@@_info:n { \msg_info:nn {fontspec} }
8 \cs_new:Npn \@@_info:nx { \msg_info:nxx {fontspec} }
9 \cs_new:Npn \@@_info:nxx { \msg_info:nxxx {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 {nxx}
12 \cs_generate_variant:Nn \msg_new:nnnn {nxxx}
13 \cs_new:Nn \@@_msg_new:nnn
14 { \msg_new:nxx {#1} {#2} { \tl_trim_spaces:n {#3} } }
15 \cs_new:Nn \@@_msg_new:nnnn
16 { \msg_new:nxxx {#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-ligtext}
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 '\l_fontspec_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  \l_fontspec_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 { \@@_warning:n {cm-default-obsolete} }
3 \DeclareOption{math}{\bool_set_true:N \g_@@_math_bool}
4 \DeclareOption{no-math}{\bool_set_false:N \g_@@_math_bool}
5 \DeclareOption{config}{\bool_set_true:N \g_@@_cfg_bool}
6 \DeclareOption{no-config}{\bool_set_false:N \g_@@_cfg_bool}
7 \DeclareOption{euenc}{\bool_set_true:N \g_@@_euenc_bool}
8 \DeclareOption{tuenc}{\bool_set_false:N \g_@@_euenc_bool}
9 \DeclareOption{quiet}
10 {
11 \msg_redirect_module:nnn { fontspec } { warning } { info }
12 \msg_redirect_module:nnn { fontspec } { info } { none }
13 }
14 \DeclareOption{silent}
15 {
16 \msg_redirect_module:nnn { fontspec } { warning } { none }
17 \msg_redirect_module:nnn { fontspec } { info } { none }
18 }
19 \ExecuteOptions{config,math,tuenc}
20 \ProcessOptions*
```

### 1.2 Encodings

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

```
21 \bool_if:NF \g_@@_euenc_bool
22 {
23 \file_if_exist:nTF {tuenc.def}
24 {
25 \cs_if_exist:cF {T@TU}
26 {
27 \@@_warning:n {tu-clash}
28 \DeclareFontEncoding{TU}{}{}
29 \DeclareFontSubstitution{TU}{lmr}{m}{n}
30 }
31 }
32 {
33 \@@_warning:n {tu-missing}
34 \bool_set_true:N \g_@@_euenc_bool
35 }
36 }
37 \bool_if:NTF \g_@@_euenc_bool
38 {
39 <XE> \tl_set:Nn \g_fontspec_encoding_tl {EU1}
```

```

40 <LU> \tl_set:Nn \g_fontspec_encoding_tl {EU2}
41 }
42 { \tl_set:Nn \g_fontspec_encoding_tl { TU } }

43 \tl_set:Nn \rmdefault {lmr}
44 \tl_set:Nn \sfdefault {lmss}
45 \tl_set:Nn \ttdefault {lmtt}
46 \RequirePackage[\g_fontspec_encoding_tl]{fontenc}
47 \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:

```

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

```

Dealing with a couple of the problems introduced by babel:

```

49 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
50 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
51 \AtBeginDocument
52 {
53 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
54 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
55 }

```

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.

```

56 \bool_if:NT \g_@@_euenc_bool
57 {
58 <LU> \cs_set_eq:NN \fontspec_tmp: \XeTeXpicfile
59 <LU> \cs_set:Npn \XeTeXpicfile {}
60 \RequirePackage{xunicode}
61 <LU> \cs_set_eq:NN \XeTeXpicfile \fontspec_tmp:
62 }

```

### 1.3 Generic functions

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

```

\FontspecSetCheckBoolTrue
\FontspecSetCheckBoolFalse

```

```

63 \cs_new:Npn \FontspecSetCheckBoolTrue { \bool_set_true:N \l_@@_check_bool }
64 \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

```

```

65 \cs_new:Nn \@@_keys_set_known:nnN
66 {
67 <debug> \typeout{::: Keys~set:~{#1}~{#2} }
68 \keys_set_known:nnN {#1} {#2} #3
69 <debug> \typeout{::: Leftover:~{#3} }
70 }
71 \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.

```
72 \cs_new:Nn \@@_int_mult_truncate:Nn
73 {
74   \int_set:Nn #1 { \__dim_eval:w #2 #1 \__dim_eval_end: }
75 }
```

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

## 1.4 `expl3` variants

```
76 \cs_generate_variant:Nn \int_set:Nn {Nv}
77 \cs_generate_variant:Nn \keys_set:nn {nx}
78 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
79 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
80 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
81 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
82 \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
83 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
84 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
85 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
86 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
87 \cs_generate_variant:Nn \tl_if_empty:nF {x}
88 \cs_generate_variant:Nn \tl_if_empty:nF {f}
89 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
90 \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 25](#).

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

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

```
33 \NewDocumentCommand \setromanfont { 0{} m 0{} }
34 {
35   \@@_main_setmainfont:nn {#1,#3} {#2}
36 }
```

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

```
37 \NewDocumentCommand \newfontfamily { m O{} m O{} }
38 {
39   \@@_main_newfontfamily:nnn {#1} {#2,#4} {#3}
40 }
41 \NewDocumentCommand \newfontface { m O{} m O{} }
42 {
43   \@@_main_newfontface:nnn {#1} {#2,#4} {#3}
44 }
45 \NewDocumentCommand \defaultfontfeatures { t+ o m }
46 {
47   \@@_main_defaultfontfeatures:nnn {#1} {#2} {#3}
48 }
49 \NewDocumentCommand \addfontfeatures {m}
50 {
51   \@@_main_addfontfeatures:n {#1}
52 }
53 \NewDocumentCommand \addfontfeature {m}
54 {
55   \@@_main_addfontfeatures:n {#1}
56 }
57 \NewDocumentCommand \newfontfeature {mm}
58 {
59   \@@_main_newfontfeature:nn {#1} {#2}
60 }
61 \NewDocumentCommand \newAATfeature {mmmm}
62 {
63   \@@_main_newAATfeature:nnnn {#1} {#2} {#3} {#4}
64 }
65 \NewDocumentCommand \newopentypefeature {mmm}
66 {
67   \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
68 }
```

`\newICUfeature` Deprecated.

```
69 \NewDocumentCommand \newICUfeature {mmm}
70 {
71   \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
72 }
```

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

```
73 \NewDocumentCommand \aliasfontfeature {mm}
74 {
75   \@@_main_aliasfontfeature:nn {#1} {#2}
76 }
77 \NewDocumentCommand \aliasfontfeatureoption {mmm}
78 {
79   \@@_main_aliasfontfeatureoption:nnn {#1} {#2} {#3}
80 }
```

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

```
81 \NewDocumentCommand \newfontscript {mm}
82 {
83   \fontspec_new_script:nn {#1} {#2}
84 }
```

*(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.

```
85 \NewDocumentCommand \newfontlanguage {mm}
86 {
87   \fontspec_new_lang:nn {#1} {#2}
88 }
```

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

```
89 \NewDocumentCommand \DeclareFontsExtensions {m}
90 {
91   \@@_main_DeclareFontsExtensions:n {#1}
92 }
93 \NewDocumentCommand \IfFontFeatureActiveTF {mmm}
94 {
95   \@@_main_IfFontFeatureActiveTF:nnn {#1} {#2} {#3}
96 }
```

# File VIII

## fontspec-user.dtx

### 1 User command internals

#### 1.1 Font selection

`\fontspec` 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. Then this new font family is selected.

```
1 \cs_new:Nn \@@_main_fontspec:nn
2 {
3   \fontspec_set_family:Nnn \f@family {#1} {#2}
4   \fontencoding { \l_@@_nfss_enc_tl }
5   \selectfont
6   \ignorespaces
7 }
```

*(End definition for `\fontspec`. 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.

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

```

29 \use:x { \exp_not:n { \DeclareRobustCommand \sffamily }
30 {
31 \exp_not:N \fontencoding { \l_@@_nfss_enc_tl }
32 \exp_not:N \fontfamily { \g_@@_sffamily_family }
33 \exp_not:N \selectfont
34 }
35 }
36 \str_if_eq_x:nnT {\familydefault} {\sfdefault}
37 { \tl_set_eq:NN \encodingdefault \l_@@_nfss_enc_tl }
38 \@@_setsansfont_hook:nn {#1} {#2}
39 \normalfont
40 \ignorespaces
41 }

```

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

`\setmonofont` Same as above.

```

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

```

(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`.

```

59 \cs_new:Nn \@@_main_setmathrm:nn
60 {
61 <XE> \fontspec_set_family:Nnn \g_@@_mathrm_tl {#1} {#2}
62 <LU> \fontspec_set_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
63 \@@_setmathrm_hook:nn {#1} {#2}
64 }

```

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

`\setboldmathrm`

```

65 \cs_new:Nn \@@_main_setboldmathrm:nn
66 {

```

```

67 <XE> \fontspec_set_family:Nnn \g_@@_bfmathrm_t1 {#1} {#2}
68 <LU> \fontspec_set_family:Nnn \g_@@_bfmathrm_t1 {Renderer=Basic,#1} {#2}
69 \@@_setboldmathrm_hook:nn {#1} {#2}
70 }

```

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

`\setmathsf`

```

71 \cs_new:Nn \@@_main_setmathsf:nn
72 {
73 <XE> \fontspec_set_family:Nnn \g_@@_mathsf_t1 {#1} {#2}
74 <LU> \fontspec_set_family:Nnn \g_@@_mathsf_t1 {Renderer=Basic,#1} {#2}
75 \@@_setmathsf_hook:nn {#1} {#2}
76 }

```

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

`\setmathtt`

```

77 \cs_new:Nn \@@_main_setmathtt:nn
78 {
79 <XE> \fontspec_set_family:Nnn \g_@@_mathtt_t1 {#1} {#2}
80 <LU> \fontspec_set_family:Nnn \g_@@_mathtt_t1 {Renderer=Basic,#1} {#2}
81 \@@_setmathtt_hook:nn {#1} {#2}
82 }

```

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

Hooks:

```

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

```

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

```

90 \onlypreamble\setmathrm
91 \onlypreamble\setboldmathrm
92 \onlypreamble\setmathsf
93 \onlypreamble\setmathtt

```

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

```

94 \tl_set:Nn \g_@@_mathrm_t1 {\rmdefault}
95 \tl_set:Nn \g_@@_mathsf_t1 {\sfdefault}
96 \tl_set:Nn \g_@@_mathtt_t1 {\ttdefault}

```

`\newfontfamily` This macro takes the arguments of `\fontspec` with a prepended *(instance cmd)*. This command is used when a specific font instance needs to be referred to repetitively (*e.g.*, in a section heading) since continuously calling `\fontspec_select:nn` is inefficient because it must parse the option arguments every time.

`\fontspec_select:nn` defines a font family and saves its name in `\l_fontspec_family_t1`. This family is then used in a typical NFSS `\fontfamily` declaration, saved in the macro name specified.

```

97 \cs_new:Nn \@@_main_newfontfamily:nnn
98 {
99   \fontspec_set_family:cnn { g_@@_ \cs_to_str:N #1 _family } {#2} {#3}
100   \use:x
101   {
102     \exp_not:N \DeclareRobustCommand \exp_not:N #1
103     {
104       \exp_not:N \fontfamily { \use:c {g_@@_ \cs_to_str:N #1 _family} }
105       \exp_not:N \fontencoding { \l_@@_nfss_enc_tl }
106       \exp_not:N \selectfont
107     }
108   }
109 }

```

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

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

```

110 \cs_new:Nn \@@_main_newfontface:nnn
111 {
112   \newfontfamily #1 [ BoldFont={},ItalicFont={},SmallCapsFont={},#2 ] {#3}
113 }

```

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

## 1.2 Font feature selection

`\defaultfontfeatures` This macro takes one argument that consists of all of feature options that will be applied by default to all subsequent `\fontspec`, et al., commands. It stores its value in `\g_fontspec_default_fontopts_tl` (initialised empty), which is concatenated with the individual macro choices in the [...] macro.

```

114 \cs_new:Nn \@@_main_defaultfontfeatures:nnn
115 {
116   \IfNoValueTF {#2}
117   { \@@_set_default_features:nn {#1} {#3} }
118   { \@@_set_font_default_features:nnn {#1} {#2} {#3} }
119   \ignorespaces
120 }
121 \cs_new:Nn \@@_set_default_features:nn
122 {
123   \IfBooleanTF {#1} \clist_put_right:Nn \clist_set:Nn
124   \g_@@_default_fontopts_clist {#2}
125 }

```

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.

```

126 \cs_new:Nn \@@_set_font_default_features:nnn
127 {
128   \clist_map_inline:nn {#2}
129   {

```

```

130 \tl_if_single:nTF {##1}
131 { \tl_set:No \l_@@_tmp_tl { \cs:w g_@@_ \cs_to_str:N ##1 _family\cs_end: } }
132 { \@@_sanitise_fontname:Nn \l_@@_tmp_tl {##1} }
133
134 \IfBooleanTF {#1}
135 {
136   \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
137   { \tl_clear:N \l_@@_tmpb_tl }
138   \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
139   \prop_gput:NVV \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
140 }
141 {
142   \tl_if_empty:nTF {#3}
143   { \prop_gremove:NV \g_@@_fontopts_prop \l_@@_tmp_tl }
144   { \prop_put:NVn \g_@@_fontopts_prop \l_@@_tmp_tl {#3,} }
145 }
146 }
147 }

```

(End definition for `\defaultfontfeatures`. 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.

```

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

```

```

166 }
167 {
168   \@@_warning:nx {addfontfeatures-ignored} {#1}
169 }
170 \ignorespaces
171 }

```

(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.

```

172 \cs_new:Nn \@@_main_newfontfeature:nn
173 {
174   \keys_define:nn { fontspec }
175   {
176     #1 .code:n =
177     {
178       \@@_update_featstr:n {#2}
179     }
180   }
181 }

```

(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.

```

182 \cs_new:Nn \@@_main_newAATfeature:nmnn
183 {
184   \keys_if_exist:nnF { fontspec } {#1}
185   { \@@_define_aat_feature_group:n {#1} }
186
187   \keys_if_choice_exist:nmT {fontspec} {#1} {#2}
188   { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
189
190   \@@_define_aat_feature:nmnn {#1}{#2}{#3}{#4}
191 }

```

(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.

```

192 \cs_new:Nn \@@_main_newopentypefeature:nmn
193 {
194   \keys_if_exist:nnF { fontspec / options } {#1}
195   { \@@_define_opentype_feature_group:n {#1} }
196
197   \keys_if_choice_exist:nmT {fontspec} {#1} {#2}
198   { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
199 }

```

```

200 \exp_args:Nnnx \@@_define_opentype_feature:nnnnn
201   {#1} {#2} { \@@_strip_plus_minus:n {#3} } {#3} {}
202 }
203 \cs_new:Nn \@@_strip_plus_minus:n { \@@_strip_plus_minus_aux:Nq #1 \q_nil }
204 \cs_new:Npn \@@_strip_plus_minus_aux:Nq #1#2 \q_nil
205   {
206     \str_case:nnF {#1} { {+} {#2} {-} {#2} } {#1#2}
207   }

```

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

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

```

208 \cs_new:Nn \@@_main_aliasfontfeature:nn
209   {
210   <debug> \typeout{::::::::::::::::::::::::::^~J:: aliasfontfeature{#1}{#2}}
211     \bool_set_false:N \l_@@_alias_bool
212
213     \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
214       {
215         \keys_if_exist:nnT {##1} {#1}
216         {
217         <debug> \typeout{::: Key~exists~##1~/~#1}
218           \bool_set_true:N \l_@@_alias_bool
219           \keys_define:nn {##1}
220             { #2 .code:n = { \keys_set:nn {##1} { #1 = {####1} } } }
221         }
222       }
223
224     \bool_if:NF \l_@@_alias_bool
225       { \@@_warning:nx {rename-feature-not-exist} {#1} }
226   }

```

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

**\aliasfontfeatureoption**

```

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

```

```

244     \keys_define:nn { ##1 / #1 }
245     { #3Reset .code:n = { \keys_set:nn {##1} { #1 = {#2Reset} } } }
246   }
247
248   \keys_if_exist:nnT { ##1 / #1 } {#20ff}
249   {
250 <debug> \typeout{::: Keyval~exists~##1~/~#1~=#~#20ff}
251     \keys_define:nn { ##1 / #1 }
252     { #30ff .code:n = { \keys_set:nn {##1} { #1 = {#20ff} } } }
253   }
254 }
255
256 \bool_if:NF \l_@@_alias_bool
257   { \@@_warning:nx {rename-feature-not-exist} {#1/#2} }
258 }

```

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

```

\DeclareFontsExtensions dfont would never be uppercase, right?
259 \cs_new:Nn \@@_main_DeclareFontsExtensions:n
260   {
261   \clist_set:Nn \l_@@_extensions_clist { #1 }
262   \tl_remove_all:Nn \l_@@_extensions_clist {~}
263   }
264 \DeclareFontsExtensions{.otf,.ttf,.OTF,.TTF,.ttc,.TTC,.dfont}

```

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

#### \IfFontFeatureActiveTF

```

265 \cs_new:Nn \@@_main>IfFontFeatureActiveTF:nnn
266   {
267 <debug> \typeout{^^J::::::::::::::::::::::::::::::::::::::::::::::::::}
268 <debug> \typeout{:IfFontFeatureActiveTF \exp_not:n{#1}{#2}{#3}}
269   \@@_if_font_feature:nTF {#1} {#2} {#3}
270   }
271 \prg_new_conditional:Nnn \@@_if_font_feature:n {TF}
272   {
273   \tl_gclear:N \g_@@_single_feat_tl
274   \group_begin:
275     \@@_font_suppress_not_found_error:
276     \@@_init:
277     \bool_set_true:N \l_@@_ot_bool
278     \bool_set_true:N \l_@@_never_check_bool
279     \bool_set_false:N \l_@@_firsttime_bool
280     \clist_clear:N \l_@@_fontfeat_clist
281     \@@_get_features:Nn \l_@@_rawfeatures_sclist {#1}
282   \group_end:
283
284 <debug> \typeout{:::> \exp_not:N\l_@@_rawfeatures_sclist->~{\l_@@_rawfeatures_sclist}}
285 <debug> \typeout{:::> \exp_not:N\g_@@_single_feat_tl->~{\g_@@_single_feat_tl}}
286

```

```
287 \tl_if_empty:NTF \g_@@_single_feat_tl { \prg_return_false: }
288 {
289   \exp_args:NV \fontspec_if_current_feature:nTF \g_@@_single_feat_tl
290   { \prg_return_true: } { \prg_return_false: }
291 }
292 }
```

*(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:nTF` 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_tl
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_@@_iv_str_to_num:Nn \l_@@_script_int {#1}
68     \l_@@_iv_str_to_num:Nn \l_@@_language_int {#2}
69     \l_@@_check_ot_feat:NnTF \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     \l_@@_set_font_type:N \font
80     \l_bool_if:NnTF \l_@@_ot_bool
81     {
82       \l_@@_check_script:NnTF \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     \l_@@_set_font_type:N \font
93     \l_bool_if:NnTF \l_@@_ot_bool
94     {
95       \prop_get:cnN {g_@@_fontinfo_ \f@family_prop} {script-num} \l_@@_tmpf_tl
96       \int_set:Nn \l_@@_script_int {\l_@@_tmpf_tl}
97       \prop_get:cnN {g_@@_fontinfo_ \f@family_prop} {script-tag} \l_fontspec_script_tl
98     }
99     \l_@@_check_lang:NnTF \font {#1} \prg_return_true: \prg_return_false:
100   }
101   { \prg_return_false: }
102 }
103 { \prg_return_false: }
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     { \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.

Please use `\fontspec_set_family:Nnn` instead of `\@@_select_font_family:nn`, which may change in the future.

```

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

```

```

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   \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:mn {#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

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

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

Some often-used variables to know about:

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

```

1 \cs_new_protected:Nn \@@_select_font_family:nn
2 {
3 <debug>\typeout{^^J^^J:~{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:Nn \l_@@_rawfeatures_sclist {}
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 }

```

(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:

```

41 \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.

```

42 \cs_new:Nn \@@_sanitise_fontname:Nn
43 {
44   \tl_set:Nx #1 {#2}
45 <LU> \tl_remove_all:Nn #1 {~}
46 \clist_map_inline:Nn \l_@@_extensions_clist
47 {
48   \tl_if_in:NnT #1 {##1}
49   {
50     \tl_remove_once:Nn #1 {##1}
51     \tl_set:Nn \l_@@_extension_tl {##1}
52     \clist_map_break:
53   }
54 }
55 }

```

(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.

```

56 \prg_new_conditional:Nnn \@@_if_detect_external:n {T}
57 {
58 <debug> \typeout{: : @@@_if_detect_external:n { \exp_not:n {#1} } }
59 \clist_map_inline:Nn \l_@@_extensions_clist
60 {
61   \bool_set_false:N \l_@@_tmpa_bool
62   \exp_args:Nx % <- this should be handled earlier

```

```

63   \tl_if_in:nnT {#1 <= end_of_string} {##1 <= end_of_string}
64     { \bool_set_true:N \l_@@_tmpa_bool \clist_map_break: }
65   }
66   \bool_if:NTF \l_@@_tmpa_bool \prg_return_true: \prg_return_false:
67 }

```

(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.

```

68 \cs_new:Nn \@@_init_ttc:n
69 {
70   \str_if_eq_x:nnT { \str_lower_case:f {\l_@@_extension_tl} } {.ttc}
71   {
72     \@@_sanitise_fontname:Nn \l_@@_fontname_it_tl {#1}
73     \@@_sanitise_fontname:Nn \l_@@_fontname_bf_tl {#1}
74     \@@_sanitise_fontname:Nn \l_@@_fontname_bfit_tl {#1}
75   }
76 }

```

(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.

```

77 \cs_new:Nn \@@_load_external_fontoptions:Nn
78 {
79   \bool_if:NT \l_@@_fontcfg_bool
80   {
81     <debug> \typeout{: @@@_load_external_fontoptions:Nn \exp_not:N #1 {#2} }
82     \@@_sanitise_fontname:Nn #1 {#2}
83     \tl_set:Nx \l_@@_ext_filename_tl {#1.fontspec}
84     \tl_remove_all:Nn \l_@@_ext_filename_tl {~}
85     \prop_if_in:NVF \g_@@_fontopts_prop #1
86     {
87       \exp_args:No \file_if_exist:nT { \l_@@_ext_filename_tl }
88       { \file_input:n { \l_@@_ext_filename_tl } }
89     }
90   }
91 }

```

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

\@@\_extract\_all\_features:

```

92 \cs_new:Nn \@@_extract_all_features:n
93 {
94   <debug> \typeout{: @@@_extract_all_features:n { \unexpanded {#1} } }
95   \bool_if:NTF \l_@@_disable_defaults_bool
96   {
97     \clist_set:Nx \l_@@_all_features_clist {#1}
98   }
99   {
100    \prop_get:NVNF \g_@@_fontopts_prop \l_fontspeg_fontname_tl \l_@@_fontopts_clist

```

```

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

```

(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.

```

116 \cs_new:Nn \@@_preparse_features:
117 {
118 <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.

```

119
120 \@@_keys_set_known:nxN {fontspec-preparse-external}
121 { \l_@@_all_features_clist }
122 \l_@@_keys_leftover_clist
123

```

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.

```

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

```

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

\@@\_load\_font:

```

130 \cs_new:Nn \@@_load_font:
131 {
132 <debug>\typeout{: : @@_load_font}
133 <debug>\typeout{Set~ base~ font~ for~ preliminary~ analysis: \@@_construct_font_call:nn { \l_@@_fontname_up_tl }

```

```

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

```

(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.

```

143 \cs_new:Nn \@@_construct_font_call:nnnnnn
144 {
145 <XE> " \@@_fontname_wrap:n { #1 #2 #3 }
146 <LU> " \@@_fontname_wrap:n { #1 #2 } #3
147 #4 #5
148 \str_if_eq:x:nnF {#6}{ } {:#6} "
149 }

```

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

```

150 \cs_new:Nn \@@_construct_font_call:nn
151 {
152 \@@_construct_font_call:nnnnnn
153 {#1}
154 \l_@@_extension_tl
155 \l_@@_ttc_index_tl
156 \l_fontspec_renderer_tl
157 \l_@@_optical_size_tl
158 {#2}
159 }

```

(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 un-

`\@@_font_is_name:` changed 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.

```

160 \cs_new:Nn \@@_font_is_name:
161 {
162 \cs_set_eq:NN \@@_fontname_wrap:n \use:n
163 }
164 \cs_new:Nn \@@_font_is_file:
165 {
166 \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
167 }

```

(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.

```
168 \cs_new:Nn \@@_set_scriptlang:
169 {
170   \bool_if:NT \l_@@_firsttime_bool
171   {
172     \tl_if_empty:NTF \l_@@_script_name_tl
173     {
174       \@@_check_script:NnTF \l_fontspeg_font {latn}
175       {
176         \tl_set:Nn \l_@@_script_name_tl {Latin}
177         \tl_if_empty:NT \l_@@_lang_name_tl
178         {
179           \tl_set:Nn \l_@@_lang_name_tl {Default}
180         }
181         \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
182         \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
183       }
184     }
185     \@@_info:n {no-scripts}
186   }
187 }
188 {
189   \tl_if_empty:NT \l_@@_lang_name_tl
190   {
191     \tl_set:Nn \l_@@_lang_name_tl {Default}
192   }
193   \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
194   \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
195 }
196 }
197 }
```

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

**\@@\_get\_features:Nn** This macro is a wrapper for `\keys_set:nx` 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.

```
198 \cs_new:Nn \@@_get_features:Nn
199 {
200   <debug> \typeout{: @@@_get_features:Nn \exp_not:N #1 { \exp_not:n {#2} } }
201   \@@_init_fontface:
202   \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#2}
203   \l_@@_keys_leftover_clist
204   \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
205   <*XE>
206   \bool_if:NTF \l_@@_ot_bool
207   {
```

```

208 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features::~"\l_@@_keys_leftover_clist
209 % \tracingall
210 \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
211 % \EROROR
212 }
213 {
214 <debug> \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features::~"\l_@@_keys_leftover_c
215 \bool_if:nT { \l_@@_atsui_bool || \l_@@_graphite_bool }
216 { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
217 }
218 </XE>
219 <*LU>
220 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features::~"\l_@@_keys_leftover_clist
221 \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
222 </LU>
223
224 \tl_if_empty:NF \l_@@_mapping_tl
225 { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
226
227 \str_if_eq_x:nnF { \l_@@_hexcol_tl \l_@@_opacity_tl }
228 { \g_@@_hexcol_tl \g_@@_opacity_tl }
229 { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
230
231 \tl_set_eq:NN #1 \l_@@_rawfeatures_sclist
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_exist:cF {g_@@_UID_\l_@@_fontid_tl}
240 {
241 \tl_new:c {g_@@_UID_\l_@@_fontid_tl}
242 }
243
244 \tl_if_exist:NT \l_@@_nfss_fam_tl
245 {
246 \tl_set_eq:cN {g_@@_UID_\l_@@_fontid_tl} \l_@@_nfss_fam_tl
247 }
248
249 \tl_if_empty:cT {g_@@_UID_\l_@@_fontid_tl}
250 {

```

```

251 % The font name is fully expanded, in case it's defined in terms of macros, before having
252 \tl_set:Nx \l_@@_tmp_tl {#1}
253 \tl_remove_all:Nn \l_@@_tmp_tl {~}
254
255 \cs_if_exist:CTF {g_@@_family_ \l_@@_tmp_tl _int}
256 { \int_gincr:c {g_@@_family_ \l_@@_tmp_tl _int} }
257 { \int_new:c {g_@@_family_ \l_@@_tmp_tl _int} }
258
259 \tl_gset:cx {g_@@_UID_ \l_@@_fontid_tl}
260 {
261   \l_@@_tmp_tl ( \int_use:c {g_@@_family_ \l_@@_tmp_tl _int} )
262 }
263 }
264 \tl_gset:Nv \l_fontspeg_family_tl {g_@@_UID_ \l_@@_fontid_tl}
265 \cs_if_exist:CTF {g_@@_fontinfo_ \l_fontspeg_family_tl _prop}
266   \prg_return_false: \prg_return_true:
267 }

```

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

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

```

268 \cs_new:Nn \@@_save_family:nn
269 {
270   \@@_save_fontinfo:n {#2}
271   \@@_find_autofonts:
272   \DeclareFontFamily{\l_@@_nfss_enc_tl}{\l_fontspeg_family_tl}{}
273   \@@_set_faces:
274   \@@_info:nxx {defining-font} {#1} {#2}
275 }

```

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

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

```

276 \cs_new:Nn \@@_save_fontinfo:n
277 {
278   \prop_new:c {g_@@_fontinfo_ \l_fontspeg_family_tl _prop}
279   \prop_gput:cnx {g_@@_fontinfo_ \l_fontspeg_family_tl _prop} {fontname} { #1 }
280   \prop_gput:cnx {g_@@_fontinfo_ \l_fontspeg_family_tl _prop} {options} { \l_@@_all_features }
281   \prop_gput:cnx {g_@@_fontinfo_ \l_fontspeg_family_tl _prop} {fontdef}
282   {
283     \@@_construct_font_call:nn {\l_fontspeg_fontname_tl}
284     { \l_@@_pre_feat_sclist \l_@@_rawfeatures_sclist }
285   }
286   \prop_gput:cnV {g_@@_fontinfo_ \l_fontspeg_family_tl _prop} {script-num} \l_@@_script_int
287   \prop_gput:cnV {g_@@_fontinfo_ \l_fontspeg_family_tl _prop} {lang-num} \l_@@_language_int
288   \prop_gput:cnV {g_@@_fontinfo_ \l_fontspeg_family_tl _prop} {script-tag} \l_fontspeg_script
289   \prop_gput:cnV {g_@@_fontinfo_ \l_fontspeg_family_tl _prop} {lang-tag} \l_fontspeg_lang_tl
290 }

```

(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:`

```
291 \cs_new:Nn \@@_find_autofonts:
292 {
293   \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
294   {
295     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
296     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
297     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspeg_fontname_tl} {/BI}
298   }
299
300   \bool_if:NF \l_@@_nobf_bool
301   {
302     \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspeg_fontname_tl} {/B}
303   }
304
305   \bool_if:NF \l_@@_noit_bool
306   {
307     \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspeg_fontname_tl} {/I}
308   }
309
310   \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
311 }
```

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

`\@@_set_faces:`

```
312 \cs_new:Nn \@@_set_faces:
313 {
314   \@@_add_nfssfont:nmmm \mddefault \updefault \l_fontspeg_fontname_tl \l_@@_fontfeat_up_
315   \@@_add_nfssfont:nmmm \bfdefault \updefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_clist
316   \@@_add_nfssfont:nmmm \mddefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_clist
317   \@@_add_nfssfont:nmmm \mddefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_clist
318   \@@_add_nfssfont:nmmm \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit_clis
319   \@@_add_nfssfont:nmmm \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl_clis
320
321   \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nmmmm ##2 }
322 }
323 \cs_new:Nn \@@_set_faces_aux:nmmmm
324 {
325   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
326   \@@_make_font_shapes:Nmmmm \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
327 }
```

(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”).

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

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

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

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

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

### 1.2.1 Fonts

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

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

```
348 \cs_new:Nn \@@_set_font_type:N
349 {
350   \debug \typeout{: @@@_set_font_type:}
351   \*XE)
352   \bool_set_false:N \l_@@_tfm_bool
353   \bool_set_false:N \l_@@_atsui_bool
354   \bool_set_false:N \l_@@_ot_bool
355   \bool_set_false:N \l_@@_mm_bool
356   \bool_set_false:N \l_@@_graphite_bool
```

```

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

```

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

LuaTeX only supports one:

```

373 <*LU>
374   \bool_set_true:N \l_@@_ot_bool
375 </LU>
376 }

```

(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.

```

377 \cs_new:Nn \@@_set_autofont:Nnn
378 {
379   \bool_if:NF \l_@@_external_bool
380   {
381     \tl_if_empty:xF {#2}
382     {
383       \tl_if_empty:NT #1
384       {
385         \@@_if_autofont:nnTF {#2} {#3}
386         { \tl_set:Nx #1 {#2#3} }
387         { \@@_info:nx {no-font-shape} {#2#3} }
388       }

```

```

389   }
390   }
391 }
392
393 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
394 {
395   \@@_primitive_font_set:Nnn \l_tmpa_font { \@@_construct_font_call:nn {#1} {} } {\f@size pt
396   \@@_primitive_font_set:Nnn \l_tmpb_font { \@@_construct_font_call:nn {#1#2} {} } {\f@size p
397   \str_if_eq_x:nnTF { \fontname \l_tmpa_font } { \fontname \l_tmpb_font }
398   { \prg_return_false: }
399   { \prg_return_true: }
400 }

```

*(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.

```

401 \cs_new:Nn \@@_make_font_shapes:Nnnnn
402 {
403   \group_begin:
404     \@@_keys_set_known:nxN {fontspec-prepare-external} { #4 } \l_@@_leftover_clist
405     \@@_load_fontname:n {#1}
406     \@@_declare_shape:nxxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
407   \group_end:
408 }
409
410 \cs_new:Nn \@@_load_fontname:n
411 {
412   <debug> \typeout{:: \@@_load_fontname:n {#1} }
413   \@@_load_external_fontoptions:Nn \l_fontspec_fontname_tl {#1}
414   \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
415   { \clist_clear:N \l_@@_fontopts_clist }
416   \@@_primitive_font_set:Nnn \l_fontspec_font { \@@_construct_font_call:nn {\l_fontspec_fon
417   \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-found} {#1} }
418 }

```

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

```

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

```

Wrapper for \DeclareFontShape. And finally the actual font shape declaration using \l\_@@\_nfss\_tl defined above. \l\_@@\_postadjust\_tl is defined in various places to deal with things like the hyphenation character and interword spacing.

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

```

419 \cs_new:Nn \@@_declare_shape:nmmm
420 {
421 <debug>\typeout{=~ declare_shape:~{\l_fontspeg_fontname_tl}~{#1}~{#2}}
422 \tl_clear:N \l_@@_nfss_tl
423 \tl_clear:N \l_@@_nfss_sc_tl
424 \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspeg_fontname_tl
425
426 \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
427
428 \@@_declare_shapes_normal:nn {#1} {#2}
429 \@@_declare_shapes_smcaps:nn {#1} {#2}
430 \@@_declare_shape_slanted:nn {#1} {#2}
431 \@@_declare_shape_loginfo:nn {#1} {#2}
432 }
433 \cs_generate_variant:Nn \@@_declare_shape:nmmm {nxxx}

```

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

\@@\_setup\_single\_size:nn

```

434 \cs_new:Nn \@@_setup_single_size:nn
435 {
436 \tl_clear:N \l_@@_size_tl
437 \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
438
439 \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
440 \l_@@_sizing_leftover_clist
441 \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
442 <debug>\typeout{==~ size:~\l_@@_size_tl}
443
444 % "normal"
445 \@@_load_fontname:n {\l_@@_sizedfont_tl}
446 \@@_setup_nfss:Nmmm \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
447 <debug> \typeout{===~ sized~ font:~ \l_@@_sizedfont_tl}
448
449 % small caps
450 \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist
451
452 \bool_if:NF \l_@@_nosc_bool
453 {
454 \tl_if_empty:NTF \l_@@_fontname_sc_tl
455 {
456 \@@_make_smallcaps:TF
457 {
458 <debug>\typeout{====~Small~ caps~ found.}
459 \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
460 }
461 {
462 <debug>\typeout{====~Small~ caps~ not~ found.}
463 \bool_set_true:N \l_@@_nosc_bool
464 }
465 }
466 { \@@_load_fontname:n {\l_@@_fontname_sc_tl} }% local for each size

```

```

467     }
468
469     \bool_if:NF \l_@@_nosc_bool
470     {
471         \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
472         {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
473     }
474 }

```

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

\@@\_setup\_nfss:Nnnn

```

475 \cs_new:Nn \@@_setup_nfss:Nnnn
476 {
477   <debug>\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~<\l_fontspec_fontname_tl>}
478
479   \@@_get_features:Nn \l_@@_rawfeatures_sclist { #2 , #3 , #4 }
480   <debug>\typeout{====~Gathered~features:~<\l_@@_rawfeatures_sclist>}
481
482   \tl_put_right:Nx #1
483   {
484     <\l_@@_size_tl> \l_@@_scale_tl
485     \@@_construct_font_call:nn { \l_fontspec_fontname_tl }
486     { \l_@@_pre_feat_sclist \l_@@_rawfeatures_sclist }
487   }
488 }

```

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

\@@\_declare\_shapes\_normal:nn

```

489 \cs_new:Nn \@@_declare_shapes_normal:nn
490 {
491   \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl} {\l_fontspec_family_tl}
492   {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
493 }

```

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

\@@\_declare\_shapes\_smcaps:nn

```

494 \cs_new:Nn \@@_declare_shapes_smcaps:nn
495 {
496   \tl_if_empty:NF \l_@@_nfss_sc_tl
497   {
498     \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl} {\l_fontspec_family_tl} {#1}
499     { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
500   }
501 }
502
503 \cs_new:Nn \@@_combo_sc_shape:n
504 {
505   \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
506   { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
507   { \scdefault }
508 }

```

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

`\@@_DeclareFontShape:nnnnnn`

```
509 \cs_new:Nn \@@_DeclareFontShape:nnnnnn
510 {
511 <debug>\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
512 \group_begin:
513 \normalsize
514 \cs_undefine:c {#1/#2/#3/#4/\f@size}
515 \group_end:
516 \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
517 }
518 \cs_generate_variant:Nn \@@_DeclareFontShape:nnnnnn {xxxxxx}
```

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

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

`\@@_declare_shape_slanted:nn`

```
519 \cs_new:Nn \@@_declare_shape_slanted:nn
520 {
521 \bool_if:nT
522 {
523 \str_if_eq_x_p:nn {#2} {\itdefault} &&
524 !(\str_if_eq_x_p:nn {\itdefault} {\sldefault})
525 }
526 {
527 \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl}{\l_fontspeg_family_tl}{#1}{\sldefault}
528 {<->ssub*\l_fontspeg_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
529 }
530 }
```

Lastly some informative messaging.

`\@@_declare_shape_loginfo:nn`

```
531 \cs_new:Nn \@@_declare_shape_loginfo:nn
532 {
533 \tl_gput_right:Nx \l_fontspeg_defined_shapes_tl
534 {
535 \exp_not:n { \ }
536 -- \exp_not:N \str_case:nn {#1/#2}
537 {
538 {\mddefault/\updefault} {'normal'~}
539 {\bfdefault/\updefault} {'bold'~}
540 {\mddefault/\itdefault} {'italic'~}
541 {\mddefault/\sldefault} {'slanted'~}
542 {\bfdefault/\itdefault} {'bold~ italic'~}
543 {\bfdefault/\sldefault} {'bold~ slanted'~}
544 } (#1/#2)~
545 with~ NFSS~ spec.:~
546 \l_@@_nfss_tl
547 \exp_not:n { \ }
```

```

548 -- \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
549 {
550   {\mddefault/\scdefault} {'small~ caps'~}
551   {\bfdefault/\scdefault} {'bold~ small~ caps'~}
552   {\mddefault/\itscdefault} {'italic~ small~ caps'~}
553   {\bfdefault/\itscdefault} {'bold~ italic~ small~ caps'~}
554   {\mddefault/\slscdefault} {'slanted~ small~ caps'~}
555   {\bfdefault/\slscdefault} {'bold~ slanted~ small~ caps'~}
556   }~( #1 / \@@_combo_sc_shape:n {#2} )~
557   with~ NFSS~ spec.:~
558   \l_@@_nfss_sc_tl
559   \tl_if_empty:FF {\l_@@_postadjust_tl}
560   {
561     \exp_not:N \ and~ font~ adjustment~ code: \exp_not:N \ \l_@@_postadjust_tl
562   }
563 }
564 }

```

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 565 \tl_set:Nn \l_@@_pre_feat_sclist
566 <*XE>
567 {
568   \bool_if:NT \l_@@_ot_bool
569   {
570     \tl_if_empty:NF \l_fontspec_script_tl
571     {
572       script = \l_fontspec_script_tl ;
573       language = \l_fontspec_lang_tl ;
574     }
575   }
576 }
577 </XE>
578 <*LU>
579 {
580   mode = \l_fontspec_mode_tl ;
581   \tl_if_empty:NF \l_fontspec_script_tl
582   {
583     script = \l_fontspec_script_tl ;
584     language = \l_fontspec_lang_tl ;
585   }
586 }
587 </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF 588 <LU>\cs_new:Nn \@@_make_smallcaps:TF
589 <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
590 {
591   \@@_check_ot_feat:NnTF \l_fontspec_font {smcp} {#1} {#2}

```

```

592 }
593 <*XE>
594 \cs_new:Nn \@@_make_smallcaps:TF
595 {
596   \bool_if:NTF \l_@@_ot_bool
597   { \@@_make_ot_smallcaps:TF {#1} {#2} }
598   {
599     \bool_if:NT \l_@@_atsui_bool
600     { \@@_make_AAT_feature_string:NnnTF \l_fontspeg_font {3}{3} {#1} {#2} }
601   }
602 }
603 </XE>

```

\l\_@@\_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
604 \cs_new:Nn \@@_update_featstr:n
605 {
606 <debug> \typeout{::: @@_update_featstr:n {#1}}
607   \bool_if:NF \l_@@_firsttime_bool
608   {
609     \tl_gset:Nx \g_@@_single_feat_tl { #1 }
610 <debug> \typeout{:::~ Adding~ feature.}
611     \tl_gput_right:Nx \l_@@_rawfeatures_sclist {#1;}
612   }
613 }

```

```

\@@_remove_clashing_featstr:n
614 \cs_new:Nn \@@_remove_clashing_featstr:n
615 {
616 <debug> \typeout{::: @@_remove_clashing_featstr:n {#1}}
617   \clist_map_inline:nn {#1}
618   {
619 <debug> \typeout{:::~ Removing~ feature~ "##1;}
620     \tl_gremove_all:Nn \l_@@_rawfeatures_sclist {##1;}
621   }
622 }

```

### 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:
623 \cs_set:Npn \@@_init:
624 {
625 <debug> \typeout{: @@_init:}
626   \bool_set_false:N \l_@@_ot_bool
627   \bool_set_true:N \l_@@_firsttime_bool
628   \@@_font_is_name:
629   \tl_clear:N \l_@@_font_path_tl
630   \tl_clear:N \l_@@_optical_size_tl
631   \tl_clear:N \l_@@_ttc_index_tl

```

```

632 \tl_clear:N \l_fontspec_renderer_tl
633 \tl_clear:N \l_fontspec_defined_shapes_tl
634 \tl_clear:N \g_@@_curr_series_tl
635 \tl_gset_eq:NN \l_@@_nfss_enc_tl \g_fontspec_encoding_tl
636
637 <*LU>
638 \tl_set:Nn \l_fontspec_mode_tl {node}
639 \int_set:Nn \luatex_prehyphenchar:D { \- } % fixme
640 \int_zero:N \luatex_posthyphenchar:D % fixme
641 \int_zero:N \luatex_preexhyphenchar:D % fixme
642 \int_zero:N \luatex_postexhyphenchar:D % fixme
643 </LU>
644 }

```

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

```

\@@_init_fontface: 645 \cs_new:Nn \@@_init_fontface:
646 {
647   \tl_clear:N \l_@@_rawfeatures_sclist
648   \tl_clear:N \l_@@_scale_tl
649   \tl_set_eq:NN \l_@@_opacity_tl \g_@@_opacity_tl
650   \tl_set_eq:NN \l_@@_hexcol_tl \g_@@_hexcol_tl
651   \tl_set_eq:NN \l_@@_postadjust_tl \g_@@_postadjust_tl
652   \tl_clear:N \l_@@_wordspace_adjust_tl
653   \tl_clear:N \l_@@_punctspace_adjust_tl
654 }

```

## 1.4 Miscellaneous

\@@\_iv\_str\_to\_num:Nn 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 -.

```

655 \cs_new:Nn \@@_iv_str_to_num:Nn
656 {
657   \@@_strip_leading_sign:Nw #1#2 \q_nil
658 }
659 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
660 {
661   \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
662     { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
663     { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
664 }
665 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
666 {
667   \int_set:Nn #1
668   {

```

```
669     `#2 * "1000000
670   + `#3 * "10000
671   + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
672   + \ifx \c_empty_tl #5 32 \else `#5 \fi
673   }
674 }
675 \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
#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)

\@@_define_opentype_feature:nmnnn 5 \cs_new:Nn \@@_feat_prop_add:n
6 {
7   \tl_if_empty:nF {#1}
8   {
9     \prop_if_in:NnF \g_@@_OT_features_prop {#1}
10    {
11      \prop_gput:Nnn \g_@@_OT_features_prop {#1} {#2}
12    }
13  }
14 }
15 \cs_new:Nn \@@_define_opentype_feature:nmnnn
16 {
17   \@@_feat_prop_add:n {#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:nmn {#3} {#4} {#5}
33       }
34     }
35   }
36 }
```

```

\@@_define_opentype_onoffreset:nnmm
#1 : Feature key
#2 : Feature option val
#3 : Check feature
#4 : Tag prefix to activate: +#4 = on, -#4 = off.
#5 : Tags to remove in the on case (clist)
37 \cs_new:Nn \@@_feat_off:n {#10ff}
38 \cs_new:Nn \@@_feat_reset:n {#1Reset}
39 \cs_new:Nn \@@_define_opentype_onoffreset:nnmm
40 {
41   \exp_args:Nnx \@@_define_opentype_feature:nnmmn {#1} {#2} {#3} {+#4} {#5}
42   \exp_args:Nnx \@@_define_opentype_feature:nnmmn {#1} { \@@_feat_off:n {#2} } {#3} {-#4} {
43   \exp_args:Nnx \@@_define_opentype_feature:nnmmn {#1} { \@@_feat_reset:n {#2} } {} {} {+#4,-
44 }

```

```

\@@_define_opentype_onreset:nnmm
#1 : Feature key
#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:nnmm
46 {
47   \exp_args:Nnx \@@_define_opentype_feature:nnmmn {#1} {#2} {#3} {#4} {#5}
48   \exp_args:Nnx \@@_define_opentype_feature:nnmmn {#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   }
99   \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
100 }
101 </XE>
102 <*LU>
103 {
104   \cs_if_eq:NNTF #1 \font
105     { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
106     { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
107   \directlua{fontspec.check_ot_script("\l_@@_tmp_tl", "#2")}
108   \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
109 }
110 </LU>
111 }

```

(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 `\@tempswatru`. `\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\XeTeXOTlanguagetag #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     \else
128       \int_incr:N \l_tmpa_int
129     \fi
130   }
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  $\emptyset$ , 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     }
170     \else
171     \int_incr:N \l_tmpa_int
172     \fi
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 {lfbid}{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 {pnum}{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 {rlig}{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 {smp1}{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 {vkrn}{Vertical~Kerning}
311 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional~Alternate~Vertical~M}
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:nmmn`

```
3 \cs_new:Nn \@@_define_aat_feature:nmmn
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:nmmn`. 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 55).

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>TeX doesn't return a selector string for this number, since the feature is defined for the 'switching on' value. So we need to check the selector of the previous number, and then prefix the feature string with ! to denote the switch.

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

```
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 is the tedious section where we correlate all possible (eventually) font feature requests with their X<sub>Y</sub>TeX representations.

```
1 \clist_set: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 }
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 }
```

### 1.0.1 Pre-parsing naming information

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:nmn {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:nmn {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:nmn {fontspec} {Extension} {}

```

### 1.0.2 Pre-parsed features

After the font name(s) have been sorted out, now need to extract any renderer/font configuration features that need to be processed before all other font features.

**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 }

```

**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 }

```

### 1.0.3 Bold/italic choosing options

The **Bold**, **Italic**, and **BoldItalic** features are for defining explicitly the bold and italic fonts used in a font family.

**Bold (NFSS) Series** By default, fontspec uses the default bold series, `\bfdefault`. We want to be able to make this extensible.

```

100 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldSeries}
101 {

```

```

102 \tl_gset:Nx \g_@@_curr_series_tl { #1 }
103 \seq_gput_right:Nx \g_@@_bf_series_seq { #1 }
104 }

```

#### Fonts Upright:

```

105 \@@_keys_define_code:nnn {fontspec-prepare-external} {UprightFont}
106 {
107 \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
108 }
109 \@@_keys_define_code:nnn {fontspec-prepare-external} {FontName}
110 {
111 \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
112 }

```

#### Bold:

```

113 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldFont}
114 {
115 \tl_if_empty:nTF {#1}
116 {
117 \bool_set_true:N \l_@@_nobf_bool
118 }
119 {
120 \bool_set_false:N \l_@@_nobf_bool
121 \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
122
123 \seq_if_empty:NT \g_@@_bf_series_seq
124 {
125 \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
126 \seq_put_right:Nx \g_@@_bf_series_seq {\bfdefault}
127 }
128 \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
129 { \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl }
130
131 <debug>\typeout{Setting~bold~font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
132
133 \prop_put:NxV \l_@@_nfss_prop
134 {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
135
136 }
137 }

```

#### Same for italic:

```

138 \@@_keys_define_code:nnn {fontspec-prepare-external} {ItalicFont}
139 {
140 \tl_if_empty:nTF {#1}
141 {
142 \bool_set_true:N \l_@@_noit_bool
143 }
144 {
145 \bool_set_false:N \l_@@_noit_bool
146 \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
147 }

```

```
148 }
```

Simpler for bold+italic & slanted:

```
149 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldItalicFont}
150 {
151   \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
152 }
153 \@@_keys_define_code:nnn {fontspec-prepare-external} {SlantedFont}
154 {
155   \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
156 }
157 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldSlantedFont}
158 {
159   \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
160 }
```

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

```
161 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
162 {
163   \tl_if_empty:nTF {#1}
164   {
165     \bool_set_true:N \l_@@_nosc_bool
166   }
167   {
168     \bool_set_false:N \l_@@_nosc_bool
169     \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
170   }
171 }
```

## Features

```
172 \@@_keys_define_code:nnn {fontspec-prepare} {UprightFeatures}
173 {
174   \clist_set:Nn \l_@@_fontfeat_up_clist {#1}
175 }
176 \@@_keys_define_code:nnn {fontspec-prepare} {BoldFeatures}
177 {
178   \clist_set:Nn \l_@@_fontfeat_bf_clist {#1}
179 }
180 % \prop_put:NxV \l_@@_nfss_prop
181 %   {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
182 }
183 \@@_keys_define_code:nnn {fontspec-prepare} {ItalicFeatures}
184 {
185   \clist_set:Nn \l_@@_fontfeat_it_clist {#1}
186 }
187 \@@_keys_define_code:nnn {fontspec-prepare} {BoldItalicFeatures}
188 {
189   \clist_set:Nn \l_@@_fontfeat_bfit_clist {#1}
190 }
191 \@@_keys_define_code:nnn {fontspec-prepare} {SlantedFeatures}
192 {
193   \clist_set:Nn \l_@@_fontfeat_sl_clist {#1}
```

```

194 }
195 \@@_keys_define_code:nnn {fontspec-prepare} {BoldSlantedFeatures}
196 {
197   \clist_set:Nn \l_@@_fontfeat_bfsl_clist {#1}
198 }

```

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

```

199 \@@_keys_define_code:nnn {fontspec} {SmallCapsFeatures}
200 {
201   \bool_if:NF \l_@@_firsttime_bool
202   {
203     \clist_set:Nn \l_@@_fontfeat_sc_clist {#1}
204   }
205 }

```

paragraphFeatures varying by size

```

206 \@@_keys_define_code:nnn {fontspec-prepare} {SizeFeatures}
207 {
208   \clist_set:Nn \l_@@_sizefeat_clist {#1}
209   \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
210 }
211 \@@_keys_define_code:nnn {fontspec-prepare-nested} {SizeFeatures}
212 {
213   \clist_set:Nn \l_@@_sizefeat_clist {#1}
214   \tl_if_empty:NT \l_@@_this_font_tl
215   { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
216 }
217 \@@_keys_define_code:nnn {fontspec-prepare-nested} {Font}
218 {
219   \tl_set:Nn \l_@@_this_font_tl {#1}
220 }
221 \@@_keys_define_code:nnn {fontspec} {SizeFeatures}
222 {
223   % dummy
224 }
225 \@@_keys_define_code:nnn {fontspec} {Font}
226 {
227   % dummy
228 }
229 \@@_keys_define_code:nnn {fontspec-sizing} {Size}
230 {
231   \tl_set:Nn \l_@@_size_tl {#1}
232 }
233 \@@_keys_define_code:nnn {fontspec-sizing} {Font}
234 {
235   \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
236 }

```

#### 1.0.4 Font-independent features

These features can be applied to any font.

**NFSS encoding** For the very brave.

```
237 \@@_keys_define_code:nmn {fontspec-preparse} {NFSSEncoding}
238 {
239   \tl_gset:Nx \l_@@_nfss_enc_tl { #1 }
240 }
```

**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.)

```
241 \@@_keys_define_code:nmn {fontspec-preparse} {NFSSFamily}
242 {
243   \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
244   \cs_undefine:c {g_@@_UID_\l_@@_fontid_tl}
245   \tl_if_exist:NT \l_fontspec_family_tl
246     { \cs_undefine:c {g_@@_fontinfo_\l_fontspec_family_tl_prop} }
247 }
```

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

```
248 \@@_keys_define_code:nmn {fontspec} {FontFace}
249 {
250   \tl_set:No \l_@@_arg_tl { \use_iii:nmn #1 }
251   \tl_set_eq:NN \l_@@_this_feat_tl \l_@@_arg_tl
252   \tl_clear:N \l_@@_this_font_tl
253   \int_compare:nT { \clist_count:N \l_@@_arg_tl = 1 }
254     {
255     <*debug>
256       \typeout{FontFace~ parsing:~ one~ clist~ item}
257     </debug>
258     \tl_if_in:NnF \l_@@_arg_tl {=}
259     {
260     <*debug>
261       \typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
262     </debug>
263     \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_tl
264     \tl_clear:N \l_@@_this_feat_tl
265     }
266   }
267
268   \@@_add_nfssfont:nnnn
269   {\use_i:nmn #1}{\use_ii:nmn #1}{\l_@@_this_font_tl}{\l_@@_this_feat_tl}
270 }
```

**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.

```
271 \@@_keys_define_code:nmn {fontspec} {Scale}
272 {
273   \str_case:nnF {#1}
274   {
275     {MatchLowercase} { \@@_calc_scale:n {5} }
276     {MatchUppercase} { \@@_calc_scale:n {8} }
277   }
```

```

277 }
278 { \tl_set:Nx \l_@@_scale_tl {#1} }
279 \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
280 }

```

`\@@_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>T<sub>E</sub>X).

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

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

```

281 \cs_new:Nn \@@_calc_scale:n
282 {
283   \group_begin:
284
285   \fontencoding { \encodingdefault }
286   \fontfamily { \rmdefault }
287   \selectfont
288
289   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
290   \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_fontspeg_font
291
292   \tl_gset:Nx \l_@@_scale_tl
293   {
294     \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
295                 \dim_to_fp:n {\l_@@_tmpb_dim} }
296   }
297
298   \@@_info:n {set-scale}
299   \group_end:
300 }

```

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

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

```

301 \cs_new:Nn \@@_set_font_dimen:NnN
302 {
303   \dim_set:Nn #1 { \fontdimen #2 #3 }
304   \dim_compare:nNnT #1 = {0pt}
305   {
306     \settoheight #1
307     {
308       \str_if_eq:nnTF {#3} {\font} \rmfamily #3
309       \int_case:nnF #2

```

```

310     {
311       {5} {x} % x-height
312       {8} {X} % cap-height
313     } {?} % "else" clause; never reached.
314   }
315 }
316 }

```

*(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.

```

317 \@@_keys_define_code:nmn {fontspec} {WordSpace}
318 {
319   \bool_if:NF \l_@@_firsttime_bool
320   { \fontspec_parse_wordspace:w #1,,,\q_stop }
321 }
322 \@@_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}.

```

323 \cs_set:Npn \fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
324 {
325   \tl_if_empty:nTF {#4}
326   {
327     \tl_set:Nn \l_@@_wordspace_adjust_tl
328     {
329       \fontdimen 2 \font = #1 \fontdimen 2 \font
330       \fontdimen 3 \font = #1 \fontdimen 3 \font
331       \fontdimen 4 \font = #1 \fontdimen 4 \font
332     }
333   }
334   {
335     \tl_set:Nn \l_@@_wordspace_adjust_tl
336     {
337       \fontdimen 2 \font = #1 \fontdimen 2 \font
338       \fontdimen 3 \font = #2 \fontdimen 3 \font
339       \fontdimen 4 \font = #3 \fontdimen 4 \font
340     }
341   }
342 }

```

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

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

```

343 \@@_keys_define_code:nmn {fontspec} {PunctuationSpace}
344 {
345   \str_case_x:nnF {#1}
346   {
347     {WordSpace}
348     {
349       \tl_set:Nn \l_@@_punctspace_adjust_tl

```

```

350     { \fontdimen 7 \font = 0 \fontdimen 2 \font }
351   }
352   {TwiceWordSpace}
353   {
354     \tl_set:Nn \l_@@_punctspace_adjust_tl
355     { \fontdimen 7 \font = 1 \fontdimen 2 \font }
356   }
357 }
358 {
359   \tl_set:Nn \l_@@_punctspace_adjust_tl
360   { \fontdimen 7 \font = #1 \fontdimen 7 \font }
361 }
362 }
363 \@@_aff_error:n {PunctuationSpace}

```

### Secret hook into the font-adjustment code

```

364 \@@_keys_define_code:nnn {fontspec} {FontAdjustment}
365 {
366   \tl_put_right:Nx \l_@@_postadjust_tl {#1}
367 }

```

### Letterspacing

```

368 \@@_keys_define_code:nnn {fontspec} {LetterSpace}
369 {
370   \@@_update_featstr:n {letterspace=#1}
371 }

```

**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.

```

372 \@@_keys_define_code:nnn {fontspec} {HyphenChar}
373 {
374   \str_if_eq:nnTF {#1} {None}
375   {
376     \tl_put_right:Nn \l_@@_postadjust_tl
377     { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
378   }
379   {
380     \@@_warning:nx {only-xetex-feature} {HyphenChar}
381
382     \tl_if_single:nTF {#1}
383     { \tl_set:Nn \l_fontspeg_hyphenchar_tl {`#1} }
384     { \tl_set:Nn \l_fontspeg_hyphenchar_tl { #1} }
385
386     \@@_primitive_font_glyph_if_exist:NnTF \l_fontspeg_font { \l_fontspeg_hyphenchar_tl }
387     {
388       \tl_put_right:Nn \l_@@_postadjust_tl
389       { \@@_primitive_font_set_hyphenchar:Nn \font { \l_fontspeg_hyphenchar_tl } }

```

```

390     }
391     { \@@_error:nx {no-glyph}{#1} }
392   }
393 }
394 }
395 \@@_aff_error:n {HyphenChar}

```

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

```

396 \@@_keys_define_code:nmn {fontspec} {Color}
397 {
398   \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
399   {
400     \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
401   }
402   {
403     \int_compare:nTF { \tl_count:n {#1} == 6 }
404     { \tl_set:Nn \l_@@_hexcol_tl {#1} }
405     {
406       \int_compare:nTF { \tl_count:n {#1} == 8 }
407       { \fontspec_parse_colour:viii #1 }
408       {
409         \bool_if:NF \l_@@_firsttime_bool
410         { \@@_warning:nx {bad-colour} {#1} }
411       }
412     }
413   }
414 }
415 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
416 {
417   \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
418   \tl_if_eq:NNF \l_@@_opacity_tl \g_@@_opacity_tl
419   {
420     \bool_if:NF \l_@@_firsttime_bool
421     { \@@_warning:nx {opa-twice-col} {#7#8} }
422   }
423   \tl_set:Nn \l_@@_opacity_tl {#7#8}
424 }
425 \aliasfontfeature{Color}{Colour}
426 \@@_keys_define_code:nmn {fontspec} {Opacity}
427 {
428   \int_set:Nn \l_@@_tmp_int {255}
429   \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
430   \tl_if_eq:NNF \l_@@_opacity_tl \g_@@_opacity_tl
431   {
432     \bool_if:NF \l_@@_firsttime_bool
433     { \@@_warning:nx {opa-twice} {#1} }
434   }
435   \tl_set:Nx \l_@@_opacity_tl
436   {
437     \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad

```

```

438     \int_to_hex:n { \l_@@_tmp_int }
439   }
440 }

```

## Mapping

```

441 ⟨*XE⟩
442 \@@_keys_define_code:nmn {fontspec-aat} {Mapping}
443 {
444   \tl_set:Nn \l_@@_mapping_tl { #1 }
445 }
446 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
447 {
448   \tl_set:Nn \l_@@_mapping_tl { #1 }
449 }
450 ⟨/XE⟩
451 ⟨*LU⟩
452 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
453 {
454   \str_if_eq:nnTF {#1} {tex-text}
455   {
456     \@@_warning:n {no-mapping-ligtext}
457     \msg_redirect_name:nnn {fontspec} {no-mapping-ligtext} {none}
458     \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
459   }
460   { \@@_warning:n {no-mapping} }
461 }
462 ⟨/LU⟩

```

### 1.0.5 Continuous font axes

```

463 \@@_keys_define_code:nmn {fontspec} {Weight}
464 {
465   \@@_update_featstr:n{weight=#1}
466 }
467 \@@_keys_define_code:nmn {fontspec} {Width}
468 {
469   \@@_update_featstr:n{width=#1}
470 }
471 \@@_keys_define_code:nmn {fontspec} {OpticalSize}
472 ⟨*XE⟩
473 {
474   \bool_if:NTF \l_@@_ot_bool
475   {
476     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
477   }
478   {
479     \bool_if:NT \l_@@_mm_bool
480     {
481       \@@_update_featstr:n { optical size = #1 }
482     }
483   }

```

```

484 \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
485 {
486   \bool_if:NT \l_@@_firsttime_bool
487     { \@@_warning:n {no-opticals} }
488   }
489 }
490 </XE>
491 <*LU>
492 {
493   \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
494 }
495 </LU>

```

### 1.0.6 Font transformations

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

```

496 \keys_define:nn {fontspec}
497 {
498   FakeSlant .code:n =
499     {
500       \@@_update_featstr:n{slant=#1}
501     },
502   FakeSlant .default:n = {0.2}
503 }
504 \keys_define:nn {fontspec}
505 {
506   FakeStretch .code:n =
507     {
508       \@@_update_featstr:n{extend=#1}
509     },
510   FakeStretch .default:n = {1.2}
511 }
512 <*XE>
513 \keys_define:nn {fontspec}
514 {
515   FakeBold .code:n =
516     {
517       \@@_update_featstr:n {embolden=#1}
518     },
519   FakeBold .default:n = {1.5}
520 }
521 </XE>
522 <*LU>
523 \keys_define:nn {fontspec}
524 {
525   FakeBold .code:n = { \@@_warning:n {fakebold-only-xetex} }
526 }
527 </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.)

```

528 \keys_define:nn {fontspec}
529 {
530   AutoFakeSlant .code:n =
531   {
532     \bool_if:NT \l_@@_firsttime_bool
533     {
534       \tl_set:Nn \l_@@_fake_slant_tl {#1}
535       \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
536       \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
537       \bool_set_false:N \l_@@_noit_bool
538
539       \tl_if_empty:NF \l_@@_fake_embolden_tl
540       {
541         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
542           {FakeBold=\l_@@_fake_embolden_tl}
543         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
544         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
545       }
546     }
547   },
548   AutoFakeSlant .default:n = {0.2}
549 }

```

Same but reversed:

```

550 \keys_define:nn {fontspec}
551 {
552   AutoFakeBold .code:n =
553   {
554     \bool_if:NT \l_@@_firsttime_bool
555     {
556       \tl_set:Nn \l_@@_fake_embolden_tl {#1}
557       \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
558       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
559       \bool_set_false:N \l_@@_nobf_bool
560
561       \tl_if_empty:NF \l_@@_fake_slant_tl
562       {
563         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
564           {FakeSlant=\l_@@_fake_slant_tl}
565         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
566         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
567       }
568     }
569   },
570   AutoFakeBold .default:n = {1.5}
571 }

```

### 1.0.7 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.

```
572 \@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}  
573 {  
574   \@@_update_featstr:n {#1}  
575 }  
576 \@@_keys_define_code:nnn {fontspec-aat} {RawFeature}  
577 {  
578   \@@_update_featstr:n {#1}  
579 }
```

## 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,+rliq,-rliq,+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}      {rliq} {rliq} {}
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,+uni
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   +sups,-sups,
63   +subs,-subs,
64   +ordn,-ordn,
65   +numr,-numr,
66   +dnom,-dnom,
67   +sinf,-sinf,
68 }
69 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Superior} {sups} {sups} {+
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 +csw, -csw,
79 +calt, -calt,
80 +init, -init,
81 +fina, -fina,
82 +falt, -falt,
83 +medi, -medi,
84 }
85 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Swash} {csw} {csw} {}
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,+jp8}
162 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Simplified} {smpl} {smpl} {+trad,+jp78,+jp8}
163 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978} {jp78} {jp78} {+trad,+smpl,+jp8}
164 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983} {jp83} {jp83} {+trad,+smpl,+jp7}
165 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990} {jp90} {jp90} {+trad,+smpl,+jp7}

```

```

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 vkrn 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} {vkrn} {vkrn} {}
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 = {0} ,
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     \l_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     \l_warning:nx {language-not-exist} {#1}
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       \l_warning:nx {language-not-exist} {#1}
310       \keys_set:nn { fontspec-opentype } { Language = Default }
311     }
312   }
313 }

```

### Default

```

314 \l_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     \l_warning:nx {language-not-exist} {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       \l_warning:nx {language-not-exist} {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}{cyrl}
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}{syrc}
121 \newfontscript{Tagalog}{tglg}
122 \newfontscript{Tagbanwa}{tagb}
123 \newfontscript{Tai~Le}{tale}
124 \newfontscript{Tai~Lu}{talu}
125 \newfontscript{Tai~Tham}{lana}
126 \newfontscript{Tai~Viet}{tavn}
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}{tibn}
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}{hani}
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{Bhojpurī}{BHO}
31 \newfontlanguage{Bikol}{BIK}
32 \newfontlanguage{Bilen}{BIL}
33 \newfontlanguage{Blackfoot}{BKF}
34 \newfontlanguage{Balochi}{BLI}
35 \newfontlanguage{Balante}{BLN}
36 \newfontlanguage{Balti}{BLT}
37 \newfontlanguage{Bambara}{BMB}
38 \newfontlanguage{Bamileke}{BML}
39 \newfontlanguage{Breton}{BRE}
40 \newfontlanguage{Brahui}{BRH}
41 \newfontlanguage{Braj~Bhasha}{BRI}
42 \newfontlanguage{Burmese}{BRM}
43 \newfontlanguage{Bashkir}{BSH}
44 \newfontlanguage{Beti}{BTI}
```

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

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

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

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

249 \newfontlanguage{Naga-Assamese}{NAG}  
250 \newfontlanguage{Nanai}{NAN}  
251 \newfontlanguage{Naskapi}{NAS}  
252 \newfontlanguage{N-Cree}{NCR}  
253 \newfontlanguage{Ndebele}{NDB}  
254 \newfontlanguage{Ndonga}{NDG}  
255 \newfontlanguage{Nepali}{NEP}  
256 \newfontlanguage{Newari}{NEW}  
257 \newfontlanguage{Nagari}{NGR}  
258 \newfontlanguage{Norway-House-Cree}{NHC}  
259 \newfontlanguage{Nisi}{NIS}  
260 \newfontlanguage{Niuean}{NIU}  
261 \newfontlanguage{Nkole}{NKL}  
262 \newfontlanguage{N'ko}{NKO}  
263 \newfontlanguage{Dutch}{NLD}  
264 \newfontlanguage{Nogai}{NOG}  
265 \newfontlanguage{Norwegian}{NOR}  
266 \newfontlanguage{Northern-Sami}{NSM}  
267 \newfontlanguage{Northern-Tai}{NTA}  
268 \newfontlanguage{Esperanto}{NTO}  
269 \newfontlanguage{Nynorsk}{NYN}  
270 \newfontlanguage{Oji-Cree}{OCR}  
271 \newfontlanguage{Ojibway}{OBJ}  
272 \newfontlanguage{Oriya}{ORI}  
273 \newfontlanguage{Oromo}{ORO}  
274 \newfontlanguage{Ossetian}{OSS}  
275 \newfontlanguage{Palestinian-Aramaic}{PAA}  
276 \newfontlanguage{Pali}{PAL}  
277 \newfontlanguage{Punjabi}{PAN}  
278 \newfontlanguage{Palpa}{PAP}  
279 \newfontlanguage{Pashto}{PAS}  
280 \newfontlanguage{Polytonic-Greek}{PGR}  
281 \newfontlanguage{Pilipino}{PIL}  
282 \newfontlanguage{Palaung}{PLG}  
283 \newfontlanguage{Polish}{PLK}  
284 \newfontlanguage{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{Tuvina}{TUV}
356 \newfontlanguage{Twi}{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>3</sub>TeX.

### 1.1 Ligatures

```
1 \@@_define_aat_feature_group:n {Ligatures}
2 \@@_define_aat_feature:n:n:n {Ligatures} {Required} {1} {0}
3 \@@_define_aat_feature:n:n:n {Ligatures} {NoRequired} {1} {1}
4 \@@_define_aat_feature:n:n:n {Ligatures} {Common} {1} {2}
5 \@@_define_aat_feature:n:n:n {Ligatures} {NoCommon} {1} {3}
6 \@@_define_aat_feature:n:n:n {Ligatures} {Rare} {1} {4}
7 \@@_define_aat_feature:n:n:n {Ligatures} {NoRare} {1} {5}
8 \@@_define_aat_feature:n:n:n {Ligatures} {Discretionary} {1} {4}
9 \@@_define_aat_feature:n:n:n {Ligatures} {NoDiscretionary} {1} {5}
10 \@@_define_aat_feature:n:n:n {Ligatures} {Logos} {1} {6}
11 \@@_define_aat_feature:n:n:n {Ligatures} {NoLogos} {1} {7}
12 \@@_define_aat_feature:n:n:n {Ligatures} {Rebus} {1} {8}
13 \@@_define_aat_feature:n:n:n {Ligatures} {NoRebus} {1} {9}
14 \@@_define_aat_feature:n:n:n {Ligatures} {Diphthong} {1} {10}
15 \@@_define_aat_feature:n:n:n {Ligatures} {NoDiphthong} {1} {11}
16 \@@_define_aat_feature:n:n:n {Ligatures} {Squared} {1} {12}
17 \@@_define_aat_feature:n:n:n {Ligatures} {NoSquared} {1} {13}
18 \@@_define_aat_feature:n:n:n {Ligatures} {AbbrevSquared} {1} {14}
19 \@@_define_aat_feature:n:n:n {Ligatures} {NoAbbrevSquared} {1} {15}
20 \@@_define_aat_feature:n:n:n {Ligatures} {Icelandic} {1} {32}
21 \@@_define_aat_feature:n:n:n {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:n:n:n {Letters} {Normal} {3} {0}
31 \@@_define_aat_feature:n:n:n {Letters} {Uppercase} {3} {1}
32 \@@_define_aat_feature:n:n:n {Letters} {Lowercase} {3} {2}
33 \@@_define_aat_feature:n:n:n {Letters} {SmallCaps} {3} {3}
34 \@@_define_aat_feature:n:n:n {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 <XE>\providecommand\UnicodeFontTeXLigatures[mapping=tex-text;}
4 <LU>\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.

```

64 \UndeclareAccent \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 {
3   \bool_set_true:N \g_@@_pkg_euler_loaded_bool
4 }
5 {
6   \bool_set_false:N \g_@@_pkg_euler_loaded_bool
7 }
8 \cs_new:Nn \fontspec_setup_maths:
9 {
10  \ifpackageloaded{euler}
11  {
12    \bool_if:NTF \g_@@_pkg_euler_loaded_bool
13    { \bool_set_true:N \g_@@_math_euler_bool }
14    { \@@_error:n {euler-too-late} }
15  }
16  {}
17  \@ifpackageloaded{lucbmath}{\bool_set_true:N \g_@@_math_lucida_bool}{}
18  \@ifpackageloaded{lucidabr}{\bool_set_true:N \g_@@_math_lucida_bool}{}
19  \@ifpackageloaded{lucimatx}{\bool_set_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 `ETEX'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 `EulerFractur`, but it's *ugly*. So I ignore it. Sorry if this causes inconvenience.)

```
20 \DeclareSymbolFont{legacymaths}{OT1}{cmr}{m}{n}
21 \SetSymbolFont{legacymaths}{bold}{OT1}{cmr}{bx}{n}
22 \DeclareMathAccent{\acute} {\mathalpha}{legacymaths}{19}
23 \DeclareMathAccent{\grave} {\mathalpha}{legacymaths}{18}
24 \DeclareMathAccent{\ddot} {\mathalpha}{legacymaths}{127}
25 \DeclareMathAccent{\tilde} {\mathalpha}{legacymaths}{126}

```

```

26 \DeclareMathAccent{\bar}      {\mathalpha}{legacymaths}{22}
27 \DeclareMathAccent{\breve}   {\mathalpha}{legacymaths}{21}
28 \DeclareMathAccent{\check}   {\mathalpha}{legacymaths}{20}
29 \DeclareMathAccent{\hat}     {\mathalpha}{legacymaths}{94} % too bad, euler
30 \DeclareMathAccent{\dot}     {\mathalpha}{legacymaths}{95}
31 \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.

```

32 \group_begin:
33 \mathchardef\@tempa="603A \relax
34 \ifx\colon\@tempa
35 \DeclareMathSymbol{\colon}{\mathpunct}{legacymaths}{58}
36 \fi
37 \group_end:

```

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

```

38 \bool_if:NF \g_@@_math_euler_bool
39 {
40 \DeclareMathSymbol{!}{\mathclose}{legacymaths}{33}
41 \DeclareMathSymbol{:}{\mathrel}{legacymaths}{58}
42 \DeclareMathSymbol{;}{\mathpunct}{legacymaths}{59}
43 \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.

```

44 \bool_if:NF \g_@@_math_lucida_bool

```

```

45 {
46   \DeclareMathSymbol{0}{\mathalpha}{legacymaths}{`0}
47   \DeclareMathSymbol{1}{\mathalpha}{legacymaths}{`1}
48   \DeclareMathSymbol{2}{\mathalpha}{legacymaths}{`2}
49   \DeclareMathSymbol{3}{\mathalpha}{legacymaths}{`3}
50   \DeclareMathSymbol{4}{\mathalpha}{legacymaths}{`4}
51   \DeclareMathSymbol{5}{\mathalpha}{legacymaths}{`5}
52   \DeclareMathSymbol{6}{\mathalpha}{legacymaths}{`6}
53   \DeclareMathSymbol{7}{\mathalpha}{legacymaths}{`7}
54   \DeclareMathSymbol{8}{\mathalpha}{legacymaths}{`8}
55   \DeclareMathSymbol{9}{\mathalpha}{legacymaths}{`9}
56   \DeclareMathSymbol{\Gamma}{\mathalpha}{legacymaths}{0}
57   \DeclareMathSymbol{\Delta}{\mathalpha}{legacymaths}{1}
58   \DeclareMathSymbol{\Theta}{\mathalpha}{legacymaths}{2}
59   \DeclareMathSymbol{\Lambda}{\mathalpha}{legacymaths}{3}
60   \DeclareMathSymbol{\Xi}{\mathalpha}{legacymaths}{4}
61   \DeclareMathSymbol{\Pi}{\mathalpha}{legacymaths}{5}
62   \DeclareMathSymbol{\Sigma}{\mathalpha}{legacymaths}{6}
63   \DeclareMathSymbol{\Upsilon}{\mathalpha}{legacymaths}{7}
64   \DeclareMathSymbol{\Phi}{\mathalpha}{legacymaths}{8}
65   \DeclareMathSymbol{\Psi}{\mathalpha}{legacymaths}{9}
66   \DeclareMathSymbol{\Omega}{\mathalpha}{legacymaths}{10}
67   \DeclareMathSymbol{+}{\mathbin}{legacymaths}{43}
68   \DeclareMathSymbol{=}{\mathrel}{legacymaths}{61}
69   \DeclareMathDelimiter{()}{\mathopen}{legacymaths}{40}{largesymbols}{0}
70   \DeclareMathDelimiter{)}{\mathclose}{legacymaths}{41}{largesymbols}{1}
71   \DeclareMathDelimiter{[}{\mathopen}{legacymaths}{91}{largesymbols}{2}
72   \DeclareMathDelimiter{]}{\mathclose}{legacymaths}{93}{largesymbols}{3}
73   \DeclareMathDelimiter{/}{\mathord}{legacymaths}{47}{largesymbols}{14}
74   \DeclareMathSymbol{\mathdollar}{\mathord}{legacymaths}{36}
75 }
76 }

```

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

Since L<sup>A</sup>T<sub>E</sub>X 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`.

```

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

```

```

89 {
90   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g_@@_bfmmathrm_tl\mddefault\updefault
91   \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g_@@_bfmmathrm_tl\bfdefault\updefault
92   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_bfmmathrm_tl\mddefault\itdefault
93 }
94 \SetMathAlphabet\mathsf{bold}\g_fontspec_encoding_tl\g_@@_mathsf_tl\bfdefault\updefault
95 \SetMathAlphabet\mathtt{bold}\g_fontspec_encoding_tl\g_@@_mathtt_tl\bfdefault\updefault
96 }

```

(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.

```

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

```

```
131 \@@_info:n {setup-math}  
132 \fontspec_setup_maths:  
133 }  
134 }  
135 \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_clear:N \g_@@_em_prop
61     \int_zero:N \l_@@_emdef_int
62     \bool_set_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_set_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_
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}
\emminnershape 111 \cs_set:Npn \emreset { \upshape }
\emreset 112 \cs_set:Npn \emshape { \itshape }
113 \cs_set:Npn \emminnershape { \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_clear: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  $\LaTeX$  2 $\epsilon$  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  $\LaTeX$ '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 $\sim 20$ \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  $\LaTeX$ .

```
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, 639
@@ commands:	
@@_DeclareFontShape:nnnnnn	491, 498, 509, 527
\g_@@_OT_features_prop	9, 11, 64
@@_add_nfssfont:nnnn	268, 314, 315, 316, 317, 318, 319, 334
@@_aff_error:n	11, 322, 363, 395
\l_@@_alias_bool	21, 211, 218, 224, 229, 236, 256
\l_@@_all_features_clist	21, 48, 97, 107, 121, 187, 280
\g_@@_all_keyval_modules_clist	1, 42, 213, 231
\g_@@_all_opentype_feature_names_prop	65, 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_tl	250, 251, 253, 258, 263
\l_@@_atsui_bool	7, 10, 215, 353, 360, 599
\l_@@_basename_tl	10, 40, 90, 331
\g_@@_bf_series_seq	40, 103, 123, 126
\g_@@_bfmathrm_tl	67, 68, 85, 90, 91, 92, 100
@@_calc_scale:n	275, 276, 281
\g_@@_cfg_bool	1, 5, 6, 16
\l_@@_check_bool	5, 63, 64, 162, 167, 173, 189
\l_@@_check_feat_bool	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, 591
@@_check_script:Nn	79
@@_check_script:NnTF	79, 82, 174, 268, 285
@@_combo_sc_shape:n	499, 503, 548, 556
@@_construct_font_call:nn	133, 135, 138, 140, 143, 169, 283, 395, 396, 416, 485
@@_construct_font_call:nnnnnn	143, 152
\l_@@_curr_bfname_tl	121, 129, 131, 134, 181
\l_@@_curr_fontname_tl	91, 325, 326
\g_@@_curr_series_tl	96, 102, 125, 128, 131, 134, 181, 634
@@_declare_shape:nnnn	406, 419
@@_declare_shape_loginfo:nn	431, 531
@@_declare_shape_slanted:nn	430, 519
@@_declare_shapes_normal:nn	428, 489
@@_declare_shapes_smcaps:nn	429, 494
\g_@@_default_fontopts_clist	41, 109, 124
@@_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, 190
@@_define_aat_feature_group:n	1, 1, 29, 35, 44, 55, 59, 64, 68, 80, 91, 101, 108, 116, 125, 185

\@@_define_opentype_feature:nmnnn	\@@_feat_off:n . . . . . 37, 42
. . 5, 7, 28, 41, 41, 42, 43, 47, 48, 60,	\@@_feat_prop_add:nn . 1, 2, 3, 4, 5, 5, 17
76, 92, 104, 110, 111, 112, 114, 119,	\@@_feat_reset:n . . . . . 38, 43, 48
120, 121, 124, 147, 148, 150, 170, 200	\@@_find_autofonts: . . . . . 271, 291
\@@_define_opentype_feature_-	\l_@@_firsttime_bool . . . . .
group:n . . . . . 1, 6,	. . . . . 1, 29, 170, 201, 279, 319,
27, 40, 59, 75, 91, 103, 113, 123, 149,	409, 420, 432, 486, 532, 554, 607, 627
169, 187, 195, 196, 209, 222, 243, 252	\@@_font_is_file: . . . . . 30, 160, 168
\@@_define_opentype_onoffreset:nmnnn	\@@_font_is_name: . . . . . 160, 628
. . . . . 13, 14, 15, 16, 17, 18,	\l_@@_font_path_tl . . . . . 29, 89, 166, 629
33, 34, 35, 36, 37, 37, 38, 39, 50, 51,	\@@_font_suppress_not_found_-
52, 53, 54, 58, 69, 70, 71, 72, 73, 74,	error: . . . . . 5, 9, 24, 275
85, 86, 87, 88, 89, 90, 99, 100, 101,	\l_@@_fontcfg_bool . . . . . 11, 12, 18, 22, 79
102, 109, 122, 137, 138, 139, 140,	\l_@@_fontfeat_bf_clist . . . . .
141, 142, 143, 144, 145, 146, 161,	. . . . . 55, 178, 315, 557
162, 163, 164, 165, 166, 167, 168,	\l_@@_fontfeat_bfit_clist . . . . .
180, 181, 182, 183, 184, 185, 186,	. . . . . 57, 189, 318, 541, 543, 563, 565
188, 189, 190, 191, 192, 193, 194, 195	\l_@@_fontfeat_bfsl_clist 59, 197, 319
\@@_define_opentype_onreset:nmnnn	\l_@@_fontfeat_clist 52, 128, 202, 280
. . . . . 26, 45	\l_@@_fontfeat_curr_clist . . . . .
\l_@@_defining_encoding_bool . 11,	. . . . . 53, 450, 459, 472
17, 23, 24, 29, 35, 59, 61, 66, 72, 78, 84	\l_@@_fontfeat_it_clist . . . . .
\l_@@_disable_defaults_bool 20, 95, 157	. . . . . 56, 185, 316, 535
\l_@@_em_int . . . . .	\l_@@_fontfeat_sc_clist . 60, 203, 450
. 31, 93, 96, 97, 100, 102, 105, 156, 159	\l_@@_fontfeat_sl_clist . 58, 193, 317
\g_@@_em_normalise_slant_bool . .	\l_@@_fontfeat_up_clist . . . . .
. . . . . 26, 62, 68, 88	. . . . . 54, 174, 209, 314
\g_@@_em_prop . . . 60, 66, 77, 78, 94, 102	\l_@@_fontid_tl . . . . . 21, 23, 40,
\l_@@_em_switch_tl . . . . . 102, 103	86, 237, 239, 241, 244, 246, 249, 259, 264
\l_@@_em_tmp_tl . . . . . 94, 96	\l_@@_fontname_bf_tl . . . . .
\l_@@_emdef_int . . . . . 32, 61, 77, 78, 79	. . . . . 73, 107, 129, 296, 302, 315, 558
\l_@@_emshape_query_tl . 86, 90, 94, 97	\l_@@_fontname_bfit_tl . . . . . 74,
\@@_error:n . . . . . 1, 14, 441	109, 151, 295, 296, 297, 318, 544, 566
\@@_error:nn . . . . . 2, 3, 12, 14, 18,	\l_@@_fontname_bfsl_tl . . . . .
24, 30, 36, 67, 73, 79, 85, 136, 391, 417	. . . . . 111, 159, 310, 319
\g_@@_euenc_bool 7, 8, 18, 21, 34, 37, 56	\l_@@_fontname_it_tl . . . . .
\l_@@_ext_filename_tl 83, 84, 87, 88, 88	. . . . . 72, 108, 146, 295, 307, 316, 536
\l_@@_extension_tl . . . . .	\l_@@_fontname_sc_tl 112, 169, 454, 466
. . . . . 39, 45, 51, 70, 87, 90, 154	\l_@@_fontname_sl_tl 110, 155, 310, 317
\l_@@_extensions_clist . . . . .	\l_@@_fontname_tl . . . 98, 156, 158, 162
. . . . . 45, 46, 59, 261, 262	\l_@@_fontname_up_tl 9, 40, 43, 106,
\l_@@_external_bool . . . 22, 28, 40, 379	107, 111, 124, 133, 135, 136, 138, 140
\@@_extract_all_features: . . . . . 92	\@@_fontname_wrap:n . . . . .
\@@_extract_all_features:n . . . 20, 92	. . . . . 44, 145, 146, 162, 166
\l_@@_fake_embolden_tl . . . . .	\l_@@_fontopts_clist . . . . .
. . . . . 105, 539, 542, 556	. . . . . 46, 100, 101, 111, 406, 414, 415
\l_@@_fake_slant_tl . 104, 534, 561, 564	\g_@@_fontopts_prop . . . . .
\l_@@_family_fontopts_clist . . . . .	61, 85, 100, 103, 136, 139, 143, 144, 414
. . . . . 47, 103, 104, 110	\@@_get_features:Nn 28, 57, 198, 281, 479
\l_@@_family_label_tl . . . . .	\l_@@_graphite_bool . 10, 215, 356, 369
. . . . . 103, 103, 105, 152, 159	\g_@@_hexcol_tl . . . . . 118, 120, 228, 650

<code>\l_@@_hexcol_tl</code> .....	<code>\@@_main_defaultfontfeatures:nnn</code> .....
..... 85, 227, 229, 400, 404, 417, 650	..... 47, 114
<code>\@@_if_autofont:nn</code> .....	<code>\@@_main_fontspec:nn</code> .....
..... 393	..... 1, 3
<code>\@@_if_autofont:nnTF</code> .....	<code>\@@_main_newAATfeature:nnnn</code> ..
..... 385	..... 63, 182
<code>\@@_if_detect_external:n</code> .....	<code>\@@_main_newfontface:nnn</code> ....
..... 56	..... 43, 110
<code>\@@_if_detect_external:nTF</code> 12, 56, 168	<code>\@@_main_newfontfamily:nnn</code> ...
<code>\@@_if_font_feature:n</code> .....	..... 39, 97
..... 271	<code>\@@_main_newfontfeature:nn</code> ...
<code>\@@_if_font_feature:nTF</code> .....	..... 59, 172
..... 269	<code>\@@_main_newopentypefeature:nnn</code> .....
<code>\@@_if_merge_shape:n</code> .....	..... 67, 71, 192
..... 25	<code>\@@_main_setboldmathrm:nn</code> ...
<code>\@@_if_merge_shape:nTF</code> .....	..... 23, 65
..... 21, 182	<code>\@@_main_setmainfont:nn</code> .....
<code>\@@_info:n</code> .....	..... 7, 8, 35
..... 7, 131, 185, 298	<code>\@@_main_setmathrm:nn</code> .....
<code>\@@_info:nn</code> .....	..... 19, 59
..... 8, 387	<code>\@@_main_setmathsf:nn</code> .....
<code>\@@_info:nnn</code> .....	..... 27, 71
..... 9, 274	<code>\@@_main_setmathtt:nn</code> .....
<code>\@@_init:</code> .....	..... 31, 77
..... 6, 167, 276, 623	<code>\@@_main_setmonofont:nn</code> .....
<code>\@@_init_fontface:</code> .....	..... 15, 42
..... 201, 645	<code>\@@_main_setsansfont:nn</code> .....
<code>\@@_init_ttc:n</code> .....	..... 11, 25
..... 17, 68	<code>\@@_make_AAT_feature:nn</code> . 7, 10, 76, 87
<code>\@@_int_mult_truncate:Nn</code> ....	<code>\@@_make_AAT_feature_string:Nnn</code> .
..... 72, 429	..... 22
<code>\@@_iv_str_to_num:Nn</code> .....	<code>\@@_make_AAT_feature_string:NnnTF</code> .....
..... 67, 68, 85, 113, 117, 160, 655	..... 12, 15, 22, 600
<code>\@@_iv_str_to_num:w</code> ....	<code>\@@_make_OT_feature:nnn</code> .....
..... 662, 663, 665	..... 32, 50, 75, 201, 205, 217, 228, 249, 258
<code>\@@_keys_define_code:nnn</code> .....	<code>\@@_make_font_shapes:Nnnnn</code> ..
..... 7, 13, 16, 20, 24, 36, 37, 46, 78, 83, 88, 95, 100, 105, 109, 113, 138, 149, 153, 157, 161, 172, 176, 183, 187, 191, 195, 199, 206, 211, 217, 221, 225, 229, 233, 237, 241, 248, 271, 314, 317, 343, 364, 368, 372, 396, 426, 442, 446, 452, 463, 467, 471, 572, 576	<code>\@@_make_ot_smallcaps:TF</code> .....
<code>\l_@@_keys_leftover_clist</code> .....	..... 588
..... 50, 122, 125, 126, 127, 203, 204, 208, 210, 214, 216, 220, 221	<code>\@@_make_smallcaps:TF</code> ..
<code>\@@_keys_set_known:nnN</code> .....	..... 456, 588, 594
..... 65, 120, 125, 127, 202, 204, 340, 404	<code>\l_@@_mapping_tl</code> .....
<code>\l_@@_lang_name_tl</code> .....	..... 22, 23, 26, 117, 224, 225, 444, 448
..... 86, 115, 136, 177, 179, 182, 189, 191, 194	<code>\g_@@_math_bool</code> .....
<code>\l_@@_language_int</code> .....	..... 3, 4, 17, 101, 103, 104, 105, 106, 107, 108, 109, 110, 113, 115, 116, 119, 121, 122, 123, 124, 125, 126, 127, 128, 129
..... 28, 45, 68, 158, 165, 287, 305, 317, 326, 333	<code>\g_@@_math_euler_bool</code> .....
<code>\l_@@_leftover_clist</code> ....	..... 13, 13, 38
..... 49, 404, 406	<code>\g_@@_math_lucida_bool</code> .....
<code>\@@_load_external_fontoptions:Nn</code> .....	..... 14, 17, 18, 19, 44
..... 18, 77, 413	<code>\g_@@_mathrm_tl</code> .....
<code>\@@_load_font:</code> .....	..... 61, 62, 77, 78, 80, 81, 84, 87, 94, 99
..... 26, 130	<code>\g_@@_mathsf_tl</code> . 73, 74, 82, 94, 95, 101
<code>\@@_load_fontname:n</code> . 405, 410, 445, 466	<code>\g_@@_mathtt_tl</code> . 79, 80, 83, 95, 96, 102
<code>\@@_main_DeclareFontsExtensions:n</code> .....	<code>\l_@@_mm_bool</code> ....
..... 91, 259	..... 9, 355, 363, 479, 484
<code>\@@_main_IfFontFeatureActiveTF:nnn</code> .....	<code>\@@_msg_new:nnn</code> .....
..... 95, 265	..... 13, 18, 23, 44, 84, 90, 94, 104, 108, 113, 117, 122, 127, 132, 138, 142, 149, 153, 157, 161, 166, 170, 175, 180, 184, 192, 196, 200, 204, 208, 213, 218
<code>\@@_main_addfontfeatures:n</code> 51, 55, 148	<code>\@@_msg_new:nnnn</code> 15, 28, 37, 48, 58, 66, 74
<code>\@@_main_aliasfontfeature:nn</code> . 75, 208	<code>\l_@@_never_check_bool</code> .....
<code>\@@_main_aliasfontfeatureoption:nnn</code> .....	..... 23, 81, 113, 149, 278
..... 79, 227	<code>\l_@@_nfss_enc_tl</code> 4, 14, 20, 31, 37, 48, 54, 95, 105, 239, 272, 491, 498, 527, 635
	<code>\l_@@_nfss_fam_tl</code> .....
	..... 243, 244, 246
	<code>\l_@@_nfss_prop</code> .....
	..... 62, 133, 180

\l_@@_nfss_sc_tl .....	\@@_remove_clashing_featstr:n ..
..... 81, 423, 471, 496, 499, 558	..... 23, 69, 614
\l_@@_nfss_tl ... 80, 422, 446, 492, 546	\g_@@_rmfamily_family . 10, 11, 15, 129
\l_@@_nfssfont_prop ..... 63, 321, 344	\@@_sanitise_fontname:Nn .....
\l_@@_nobf_bool .....	..... 8, 9, 10, 42, 72, 73, 74, 82, 132
..... 2, 26, 117, 120, 293, 300, 559	\@@_save_family:n ..... 33, 268
\l_@@_noit_bool .....	\@@_save_family_needed:n ..... 233
..... 3, 27, 142, 145, 293, 305, 537	\@@_save_family_needed:nTF ... 31, 233
\l_@@_nosc_bool 4, 165, 168, 452, 463, 469	\@@_save_fontinfo:n ..... 270, 276
\g_@@_opacity_tl .....	\l_@@_saved_fontname_tl . 92, 424, 437
..... 119, 121, 228, 418, 430, 649	\l_@@_scale_tl .....
\l_@@_opacity_tl .....	..... 83, 198, 278, 279, 292, 484, 648
.. 84, 227, 229, 418, 423, 430, 435, 649	\l_@@_script_exist_bool .....
\l_@@_optical_size_tl .....	..... 25, 265, 272, 278
..... 93, 157, 476, 493, 630	\l_@@_script_int ..... 27, 42, 67,
\l_@@_options_tl ... 97, 155, 158, 162	96, 113, 119, 124, 157, 165, 271, 286, 289
\l_@@_ot_bool .....	\l_@@_script_name_tl .....
8, 28, 39,	.. 81, 113, 136, 146, 172, 176, 181, 193
65, 80, 93, 110, 125, 140, 206, 277,	\@@_select_font_family:n ..... 1, 41, 153, 158, 160, 161
354, 366, 374, 474, 484, 568, 596, 626	\@@_set_autofont:Nnn .....
\@@_ot_compat:n ..... 344, 348, 349,	..... 295, 296, 297, 302, 307, 310, 377
350, 351, 352, 353, 354, 355, 356,	\@@_set_default_features:n ..... 117, 121
357, 358, 359, 360, 361, 362, 363, 364	\@@_set_faces: ..... 273, 312
\g_@@_pkg_euler_loaded_bool .....	\@@_set_faces_aux:n ..... 321, 323
..... 3, 6, 12, 15	\@@_set_font_default_features:n ..... 118, 126
\g_@@_postadjust_tl .... 125, 126, 651	\@@_set_font_dimen:NnN . 289, 290, 301
\l_@@_postadjust_tl .....	\@@_set_font_type:N .....
124, 366,	27, 38, 64, 79, 92, 109, 124, 137, 139, 348
376, 388, 492, 499, 528, 559, 561, 651	\@@_set_scriptlang: ..... 27, 168
\l_@@_pre_feat_sclist 128, 284, 486, 565	\@@_setboldmathrm_hook:n ..... 69, 89
\@@_preparse_features: ..... 25, 116	\@@_setmainfont_hook:n ..... 21, 83
\l_@@_prev_unicode_name_tl ... 57, 62	\@@_setmathrm_hook:n ..... 63, 86
\l_@@_primitive_font .....	\@@_setmathsf_hook:n ..... 75, 87
25, 26	\@@_setmathtt_hook:n ..... 81, 88
\@@_primitive_font_glyph_if_	\@@_setmonofont_hook:n ..... 55, 85
exist:Nn .....	\@@_setsansfont_hook:n ..... 38, 84
30	\@@_setup_nfss:Nnn ..... 446, 471, 475
\@@_primitive_font_glyph_if_	\@@_setup_single_size:n ..... 426, 434
exist:NnTF .....	\g_@@_sffamily_family . 27, 28, 32, 130
16, 30, 386	\@@_shape_merge:n 10, 11, 12, 13, 14,
\@@_primitive_font_gset:Nnn .. 1, 139	15, 16, 17, 18, 22, 28, 33, 184, 505, 506
\@@_primitive_font_if_exist:nTF	\g_@@_single_feat_tl .....
..... 21, 169	..... 58, 73, 76, 273, 273,
\@@_primitive_font_if_null_p:N .. 13	285, 287, 289, 306, 318, 328, 335, 609
\@@_primitive_font_set:Nnn .....	\l_@@_size_tl .....
..... 1, 25, 134, 395, 396, 416	..... 78, 231, 436, 441, 442, 477, 484
\@@_primitive_font_set_hyphenchar:Nn	\l_@@_sizedfont_tl 79, 235, 437, 445, 447
..... 38, 377, 389	
\l_@@_proceed_bool .....	
54, 63, 67	
\l_@@_punctspace_adjust_tl .....	
..... 122, 126, 349, 354, 359, 653	
\l_@@_rawfeatures_sclist .....	
..... 28, 56, 127, 231, 281,	
284, 284, 479, 480, 486, 611, 620, 647	



<code>\bool_set_true:N</code> . . . . .	3, 3, 5, 7, 12, 13, 17, 18, 19, 26, 27, 28, 34, 54, 55, 59, 62, 63, 64, 92, 117, 125, 142, 157, 165, 167, 218, 236, 272, 277, 278, 358, 360, 363, 366, 369, 374, 463, 627	
<code>\bool_until_do:nn</code> . . . . .	89, 122, 163	
<code>\breve</code> . . . . .	27	
<b>C</b>		
<code>\char</code> . . . . .	5, 6, 17	
char commands:		
<code>\char_set_catcode_active:n</code> . . . . .	28, 30	
<code>\char_set_catcode_ignore:n</code> . . . . .	226	
<code>\char_set_catcode_space:n</code> . . . . .	17	
<code>\check</code> . . . . .	28	
clist commands:		
<code>\clist_clear:N</code> . . . . .	101, 104, 280, 415	
<code>\clist_count:N</code> . . . . .	253	
<code>\clist_count:n</code> . . . . .	100	
<code>\clist_map_break:</code> . . . . .	52, 64, 274	
<code>\clist_map_inline:Nn</code> . . . . .	46, 59, 213, 231	
<code>\clist_map_inline:nn</code> . . . . .	74, 74, 86, 121, 128, 204, 215, 235, 266, 426, 617	
<code>\clist_new:N</code> . . . . .	41, 42, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53	
<code>\clist_put_left:Nn</code> . . . . .	459	
<code>\clist_put_right:Nn</code> . . . . .	. . . . .	
. . . . .	123, 209, 535, 541, 543, 557, 563, 565	
<code>\clist_set:Nn</code> . . . . .	. . . . .	
. . . . .	1, 44, 97, 107, 123, 174, 178, 185, 189, 193, 197, 203, 208, 213, 261, 339	
<code>\clist_set_eq:NN</code> . . . . .	450	
<code>\colon</code> . . . . .	34, 35, 112, 112	
<code>\convertcolorspec</code> . . . . .	400	
cs commands:		
<code>\cs:w</code> . . . . .	131	
<code>\cs_end:</code> . . . . .	131	
<code>\cs_generate_variant:Nn</code> . . . . .	11, 12, 71, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 156, 433, 518, 675	
<code>\cs_gset:Npn</code> . . . . .	29	
<code>\cs_if_eq:NNTF</code> . . . . .	103, 136, 179	
<code>\cs_if_exist:NTF</code> . . . . .	3, 25, 190, 255, 265, 398	
<code>\cs_if_exist_p:N</code> . . . . .	30	
<code>\cs_new:Nn</code> . . . . .	1, 1, 1, 3, 5, 7, 8, 8, 10, 10, 11, 13, 14, 15, 15, 19, 20, 25, 37, 38, 38, 39, 42, 42, 45, 50, 59, 65, 65, 68, 71, 72, 76, 77, 77, 92, 97, 97, 110, 114, 116, 121, 126, 130, 143, 148, 150, 150, 157, 159, 160, 160, 164, 168, 172, 182, 192, 198, 203, 208, 227, 259, 262, 265, 268, 276, 281, 291, 299, 301, 312, 323, 328, 334, 344, 348, 377, 401, 410, 419, 434, 475, 489, 494, 503, 509, 519, 531, 588, 589, 594, 604, 614, 645, 655	
<code>\cs_new:Npn</code> . . . . .	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 63, 64, 77, 204, 225	
<code>\cs_new_protected:Nn</code> . . . . .	1	
<code>\cs_new_protected:Npn</code> . . . . .	58, 114	
<code>\cs_set:Npn</code> . . . . .	1, 5, 9, 50, 54, 59, 60, 66, 69, 71, 80, 87, 111, 112, 113, 152, 153, 166, 323, 415, 623, 659, 665	
<code>\cs_set_eq:NN</code> . . . . .	31, 41, 58, 61, 83, 84, 84, 85, 86, 87, 88, 89, 91, 162, 173	
<code>\cs_to_str:N</code> . . . . .	99, 104, 105, 131, 138, 181	
<code>\cs_undefine:N</code> . . . . .	86, 244, 246, 514	
<code>\cyrillicencoding</code> . . . . .	49, 53, 74	
<b>D</b>		
<code>\date</code> . . . . .	56	
<code>\ddot</code> . . . . .	24	
<code>\DeclareDocumentCommand</code> . . . . .	. . . . .	
. . . . .	9, 15, 21, 27, 33, 39, 64, 70, 76, 82	
<code>\DeclareErrorFont</code> . . . . .	42	
<code>\DeclareFontEncoding</code> . . . . .	28, 41	
<code>\DeclareFontFamily</code> . . . . .	44, 272	
<code>\DeclareFontsExtensions</code> . . . . .	89, 259	
<code>\DeclareFontShape</code> . . . . .	. . . . .	
. . . . .	46, 48, 50, 51, 51, 52, 54, 516	
<code>\DeclareFontSubstitution</code> . . . . .	29, 43	
<code>\DeclareMathAccent</code> . . . . .	. . . . .	
. . . . .	22, 23, 24, 25, 26, 27, 28, 29, 30, 31	
<code>\DeclareMathDelimiter</code> . . . . .	69, 70, 71, 72, 73	
<code>\DeclareMathSymbol</code> . . . . .	. . . . .	
. . . . .	35, 40, 41, 42, 43, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 74	
<code>\DeclareOption</code> . . . . .	1, 3, 4, 5, 6, 7, 8, 9, 14	
<code>\DeclareRobustCommand</code> . . . . .	2, 4, 12, 29, 38, 43, 46, 48, 53, 83, 102, 130	
<code>\DeclareSymbolFont</code> . . . . .	20, 77	
<code>\DeclareSymbolFontAlphabet</code> . . . . .	79	
<code>\DeclareTextCommand</code> . . . . .	7, 13, 19	
<code>\DeclareTextComposite</code> . . . . .	31	
<code>\DeclareTextCompositeCommand</code> . . . . .	37	
<code>\DeclareTextFontCommand</code> . . . . .	9, 110, 151	
<code>\DeclareTextSymbol</code> . . . . .	25	
<code>\DeclareUnicodeAccent</code> . . . . .	6	
<code>\DeclareUnicodeEncoding</code> . . . . .	21, 39	
<code>\def</code> . . . . .	13, 28, 34, 42	
<code>\defaultfontfeatures</code> . . . . .	45, 114	
<code>\Delta</code> . . . . .	57	

dim commands:	
\dim_compare:nNnTF	304
\dim_new:N	37, 38, 39
\dim_set:Nn	303
\dim_to_fp:n	294, 295
dim internal commands:	
\_dim_eval:w	74
\_dim_eval_end:	74
\directlua	6, 106, 139, 182
\discretionary	4
\do	38
\dospecials	38
\dot	30
\DTX	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
E	
\else	8, 17, 36, 94, 127, 169, 671, 672
else commands:	
\else:	34
\em	83, 110
\emfontdeclare	58, 162
\emminnershape	110, 162
\emph	110
\emreset	74, 106, 110
\emshape	110, 162
\EncodingAccent	15
\EncodingCommand	9
\EncodingComposite	27
\EncodingCompositeCommand	33
\encodingdefault	20, 37, 54, 76, 285
\EncodingSymbol	21
\endinput	8, 13, 22, 29
environments:	
listingcont*	66
verbatim*	50
\ERROROR	211
etex commands:	
\etex_iffontchar:D	32
\ExecuteOptions	19
exp commands:	
\exp_after:wN	439
\exp_args:Nnnx	200
\exp_args:Nnx	41, 42, 43, 47, 48
\exp_args:No	87
\exp_args:NV	289
\exp_args:Nx	62, 69, 426
\exp_args:Nxx	176
\exp_not:N	14, 15, 16, 20, 31, 32, 33, 48, 49, 50, 81, 102, 104, 105, 106, 200, 225, 226, 229, 237, 238, 284, 285, 536, 548, 561
\exp_not:n	12, 29, 46, 52, 58, 200, 268, 535, 547
\expandafter	29
F	
\familydefault	19, 36, 53
\FancyVerbSpace	84
\fi	10, 19, 29, 36, 36, 96, 101, 113, 119, 129, 171, 364, 371, 671, 672
fi commands:	
\fi:	36
file commands:	
\file_if_exist:nTF	23, 87
\file_input:n	88
\filedate	56
\fileversion	56
\fmtname	29
\font	3, 6, 7, 9, 9, 12, 16, 27, 38, 40, 50, 64, 69, 79, 82, 92, 99, 103, 109, 114, 124, 136, 139, 161, 177, 179, 289, 308, 329, 330, 331, 337, 338, 339, 350, 355, 360, 377, 389
font commands:	
\l_fontspec_font	15, 40, 40, 60, 134, 136, 137, 139, 141, 161, 174, 268, 285, 290, 302, 324, 331, 386, 416, 417, 591, 600
\l_tmpa_font	395, 397
\l_tmpb_font	396, 397
\fontdimen	77, 77, 303, 329, 330, 331, 337, 338, 339, 350, 355, 360
\fontdimen8	76
\fontencoding	4, 14, 31, 48, 105, 285
\fontfamily	15, 27, 32, 49, 104, 165, 286
\fontname	161, 177, 397
\fontshape	7, 22, 23
\fontspec	1, 1, 27, 28, 29, 40, 123
fontspec commands:	
\fontspec_calc_scale:n	75
\fontspec_complete_fontname:Nn	107, 111, 121, 146, 151, 155, 159, 169, 235, 325, 328
\g_fontspec_default_fontopts_tl	28, 29
\l_fontspec_defined_shapes_tl	72, 190, 533, 633
\g_fontspec_encoding_tl	23, 39, 40, 42, 46, 47, 49, 50, 53, 54, 69, 77, 78, 80, 81, 82, 83, 84, 87, 90, 91, 92, 94, 95, 635







Path	24	<code>\prop_new:N</code>	61, 62, 63, 64, 65, 66, 67, 278
<code>\Phi</code>	64	<code>\prop_put:Nnn</code>	79, 80, 133, 144, 180, 344
<code>\Pi</code>	61	<code>\providecommand</code>	1, 2, 2, 3, 3, 4, 5, 6
prg commands:		<code>\ProvidesExplFile</code>	49
<code>\prg_new_conditional:Nnn</code>	1, 5, 22, 23, 25, 30, 34, 56, 60, 75, 79, 88, 105, 111, 120, 135, 147, 164, 174, 180, 233, 271, 393	<code>\ProvidesExplPackage</code>	44, 45, 46
<code>\prg_return_false:</code>	3, 13, 16, 18, 20, 26, 27, 28, 31, 35, 36, 46, 50, 53, 57, 66, 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, 266, 287, 290, 398	<code>\Psi</code>	65
<code>\prg_return_true:</code>	3, 13, 16, 28, 28, 33, 36, 49, 50, 66, 69, 82, 82, 98, 99, 107, 114, 114, 129, 131, 143, 144, 150, 170, 173, 178, 189, 196, 266, 290, 399	Q	
<code>\prg_set_conditional:Nnn</code>	13, 21	<code>\qqquad</code>	56
<code>\ProcessOptions</code>	20	quark commands:	
prop commands:		<code>\q_nil</code>	76, 77, 203, 204, 226, 238, 657, 659, 662, 663, 665
<code>\prop_clear:N</code>	60, 116	<code>\q_stop</code>	320, 323
<code>\prop_get:NnN</code>	41, 44, 47, 48, 95, 97, 127, 142, 155, 156	R	
<code>\prop_get:NnNTF</code>	83, 84, 94, 100, 102, 103, 135, 136, 143, 414	<code>\relax</code>	5, 6, 33, 36, 40, 45, 50, 55
<code>\prop_gput:Nnn</code>	11, 78, 82, 125, 139, 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, 279, 280, 280, 281, 281, 282, 283, 284, 285, 286, 286, 287, 287, 288, 288, 289, 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	<code>\RenewDocumentCommand</code>	94
<code>\prop_gput_if_new:Nnn</code>	77, 81, 124	<code>\RequirePackage</code>	5, 43, 46, 48, 60
<code>\prop_gremove:Nn</code>	143	<code>\RequirePackageWithOptions</code>	7, 12
<code>\prop_if_in:NnTF</code>	9, 85	<code>\rmddefault</code>	11, 19, 27, 43, 94, 113, 286
<code>\prop_map_inline:Nn</code>	321	<code>\rmfamily</code>	12, 76, 76, 308
		S	
		scan commands:	
		<code>\scan_stop:</code>	3, 7, 17, 32, 40, 161
		<code>\scdefault</code>	2, 3, 11, 12, 13, 14, 17, 18, 51, 505, 506, 507, 550, 551
		<code>\scshape</code>	38
		<code>\select</code>	18
		<code>\selectfont</code>	5, 7, 16, 22, 23, 33, 50, 106, 165, 287
		seq commands:	
		<code>\seq_gput_right:Nn</code>	103
		<code>\seq_if_empty:NTF</code>	123
		<code>\seq_new:N</code>	40
		<code>\seq_put_right:Nn</code>	126
		<code>\setboldmathrm</code>	21, 26, 65, 91, 113
		<code>\setmainfont</code>	5, 8, 22, 26, 111
		<code>\SetMathAlphabet</code>	80, 81, 82, 83, 87, 90, 91, 92, 94, 95
		<code>\setmathrm</code>	17, 59, 90, 113
		<code>\setmathsf</code>	25, 71, 92
		<code>\setmathtt</code>	29, 77, 93
		<code>\setmonofont</code>	13, 42
		<code>\setromanfont</code>	33
		<code>\setsansfont</code>	9, 25
		<code>\SetSymbolFont</code>	21, 78, 84
		<code>\settoheight</code>	306
		<code>\sfdefault</code>	28, 36, 44, 95
		<code>\sffamily</code>	29
		<code>\Sigma</code>	62



224, 249, 287, 342, 361, 367, 370, 383, 441, 454, 496, 539, 561, 570, 581	\ttfamily . . . . . 46
\tl_if_empty:nTF 7, 12, 18, 57, 86, 87, 88, 115, 140, 142, 163, 325, 338, 381, 559	\typeout . . . . . 3, 5, 23, 31, 37, 52, 58, 67, 69, 81, 93, 94, 97, 118, 131, 132, 133, 134, 138, 138, 150, 153, 158, 178, 200, 208, 210, 214, 217, 220, 235, 236, 237, 243, 250, 256, 261, 267, 268, 284, 285, 350, 412, 421, 442, 447, 458, 462, 477, 480, 511, 606, 610, 616, 619, 625
\tl_if_eq:NNTF . . . . . 192, 418, 430	U
\tl_if_eq:nnTF . . . . . 89, 128	\UndeclareAccent . . . . . 64
\tl_if_exist:NTF . . . . . 239, 244, 245, 505	\UndeclareCommand . . . . . 64
\tl_if_exist_p:N . . . . . 28	\UndeclareComposite . . . . . 82
\tl_if_in:NnTF . . . . . 48, 48, 258	\UndeclareSymbol . . . . . 64
\tl_if_in:nnTF . . . . . 63, 64, 66, 176	\UndeclareTextCommand . . . . . 68, 74, 80
\tl_if_single:nTF . . . . . 130, 382	\unexpanded . . . . . 94
\tl_new:N . . . . . 54, 55, 56, 57, 58, 59, 60, 68, 69, 70, 71, 72, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 122, 123, 124, 125, 127, 128, 129, 130, 131, 241	\UnicodeEncodingName . . . . . 13, 19, 25, 31, 37, 57, 58, 62, 68, 74, 80, 87
\tl_put_left:Nn . . . . . 42	\UnicodeFontFile . . . . . 1, 47, 49, 51, 53, 55
\tl_put_right:Nn 138, 366, 376, 388, 482	\UnicodeFontName . . . . . 2
\tl_remove_all:Nn 45, 84, 253, 262, 332	\UnicodeFontTeXLigatures . . . . . . . . . . 3, 4, 47, 49, 51, 53, 55
\tl_remove_once:Nn . . . . . 50	\updefault . . . . . 17, 18, 56, 77, 78, 81, 82, 83, 84, 90, 91, 94, 95, 194, 314, 315, 538, 539
\tl_replace_all:Nnn . . . . . 90, 90, 331	\upshape . . . . . 38, III, II3
\tl_set:Nn . . . . . 21, 22, 24, 26, 29, 30, 35, 38, 39, 39, 40, 42, 43, 44, 44, 45, 48, 51, 54, 58, 69, 81, 83, 86, 86, 92, 93, 94, 95, 96, 97, 98, 104, 105, 112, 120, 121, 126, 131, 137, 138, 152, 159, 176, 179, 180, 181, 187, 191, 215, 219, 231, 243, 250, 252, 270, 278, 279, 288, 304, 316, 327, 327, 330, 334, 335, 336, 349, 354, 359, 361, 367, 370, 383, 384, 386, 404, 417, 423, 435, 444, 448, 476, 493, 534, 556, 565, 638	\Upsilon . . . . . 63
\tl_set_eq:NN 11, 20, 28, 37, 45, 47, 49, 50, 53, 54, 54, 57, 62, 124, 129, 154, 162, 184, 231, 246, 251, 263, 424, 437, 536, 544, 558, 566, 649, 650, 651	\url . . . . . 40
\tl_to_str:N . . . . . 21	use commands:
\tl_to_str:n . . . . . 87, 177	\use:N . . . . . 99, 104
\tl_trim_spaces:n . . . . . 14, 16	\use:n 12, 29, 46, 100, 159, 162, 223, 439
\tl_use:N . . . . . 22, 33, 506	\use_i:nnn . . . . . 269
\l_tmpa_tl . . . . . 24, 25, 48	\use_ii:nnn . . . . . 269
\l_tmpb_tl . . . . . 30, 35, 38, 42, 45, 48	\use_iii:nnn . . . . . 250
\tmpa . . . . . 28, 29	\use_none:mn . . . . . 83, 84, 85, 86, 87, 88, 89
token commands:	\usefont . . . . . 23
\token_to_str:N . . . . . 87, 398	\UTFencname . . . . . 47, 73
\tracingall . . . . . 209	V
\ttdefault . . . . . 45, 45, 53, 96	\verb . . . . . 34, 123
	\verb* . . . . . 34, 122
	verbatim* (environment) . . . . . 50
	X
	\XeLaTeX . . . . . 34
	xetex commands:
	\xetex_suppressfontnotfounderror:D . . . . . 11
	\XeTeXcountvariations . . . . . 362
	\XeTeXfeaturename . . . . . 24
	\XeTeXfonttype . . . . . 357

<code>\XeTeXisexclusivefeature</code> .....	28	<code>\XeTeXOTlanguagetag</code> .....	124
<code>\XeTeXOTcountfeatures</code> .....	156	<code>\XeTeXOTscripttag</code> .....	91
<code>\XeTeXOTcountlanguages</code> .....	119	<code>\XeTeXpicfile</code> .....	58, 59, 61
<code>\XeTeXOTcountscripts</code> .....	86	<code>\XeTeXselectorname</code> .....	30, 35, 40
<code>\XeTeXOTfeaturetag</code> .....	165	<code>\Xi</code> .....	60