

The fontspec package

Font selection for X_YLaTeX and LuaLaTeX

WILL ROBERTSON

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

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

2019/02/12 v2.7b

Contents

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

1.3	Generic functions	19
1.4	expl3 variants	20
VI	fontspec-code-fontload.dtx	21
1	expl3 interface for primitive font loading	21
VII	fontspec-code-interfaces.dtx	23
1	User commands	23
VIII	fontspec-code-user.dtx	27
1	User command internals	27
1.1	Font selection	27
1.2	Font feature selection	30
1.3	Defining new font features	32
1.4	High level conditionals	34
1.5	\oldstylenums and \liningnums	35
IX	fontspec-code-api.dtx	36
1	Programmer's interface	36
X	fontspec-code-internal.dtx	42
1	Internals	42
1.1	The main function for setting fonts	42
1.2	Setting font shapes in a family	49
1.3	Initialisation	59
1.4	Miscellaneous	60
XI	fontspec-code-opentype.dtx	62
1	OpenType definitions code	62
1.1	Adding features when loading fonts	63
1.2	OpenType feature information	67
XII	fontspec-code-graphite.dtx	70
1	Graphite/AAT code	70

XIII	fontspec-code-keyval.dtx	72
1	Font loading (keyval) definitions	72
1.1	Pre-pre-parsing stages	72
1.2	Pre-parsed features	74
1.3	Font faces	75
1.4	General font-independent features	78
XIV	fontspec-code-feat-opentype.dtx	87
1	OpenType feature definitions	87
2	Regular key=val / tag definitions	87
2.1	Ligatures	87
2.2	Letters	87
2.3	Numbers	88
2.4	Vertical position	88
2.5	Contextuals	89
2.6	Diacritics	89
2.7	Kerning	89
2.8	Fractions	89
2.9	Style	90
2.10	CJK shape	90
2.11	Character width	91
2.12	Vertical	91
3	OpenType features that need numbering	91
3.1	Alternate	91
3.2	Variant / StylisticSet	92
3.3	CharacterVariant	92
3.4	Annotation	93
3.5	Ornament	93
4	Script and Language	93
4.1	Script	93
4.2	Language	94
5	Backwards compatibility	95
XV	fontspec-code-scripts.dtx	96
1	Font script definitions	96
XVI	fontspec-code-lang.dtx	99
1	Font language definitions	99

XVII	fontspec-code-feat-aat.dtx	107
1	AAT feature definitions	107
1.1	Ligatures	107
1.2	Letters	107
1.3	Numbers	108
1.4	Contextuals	108
1.5	Diacritics	108
1.6	Vertical position	108
1.7	Fractions	108
1.8	Alternate	108
1.9	Variant / StylisticSet	109
1.10	Style	109
1.11	CJK shape	109
1.12	Character width	110
1.13	Annotation	110
XVIII	fontspec-code-enc.dtx	111
1	Extended font encodings	111
XIX	fontspec-code-math.dtx	114
1	Selecting maths fonts	114
XX	fontspec-code-closing.dtx	119
1	Closing code	119
1.1	Finishing up	119
XXI	fontspec-code-xfss.dtx	120
1	Changes to the NFSS	120
1.1	Italic small caps and so on	120
1.2	Emphasis	121
1.3	Strong emphasis	123
XXII	fontspec-code-patches.dtx	125
1	Patching code	125
1.1	Verbatims	125
Index		128

File I

fontspec.dtx

1 Package declaration

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

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

Now exit if we're using plain T_EX; 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 {2019/02/12}{2.7b}{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   = "2.7b",
63 <lua>  date      = "2019/02/12",
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[®]TeX

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

X_ƎTeX

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

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

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

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

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

File III

fontspec-code-vars.dtx

1 Declaration of variables

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

Booleans

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

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

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

```
2 \bool_new:N \l_@@_nobf_bool
3 \bool_new:N \l_@@_noit_bool
4 \bool_new:N \l_@@_nosc_bool
5 \bool_new:N \l_@@_check_bool
6 \bool_new:N \l_@@_tfm_bool
7 \bool_new:N \l_@@_atsui_bool
8 \bool_new:N \l_@@_ot_bool
9 \bool_new:N \l_@@_mm_bool
10 \bool_new:N \l_@@_harfbuzz_bool
11 \bool_new:N \l_@@_graphite_bool
12 \bool_new:N \l_@@_fontcfg_bool
13 \bool_set_true:N \l_@@_fontcfg_bool
```

For dealing with legacy maths:

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

For package options:

```
17 \bool_new:N \g_@@_cfg_bool
18 \bool_new:N \g_@@_math_bool
19 \bool_new:N \g_@@_euenc_bool
20 \bool_new:N \l_@@_tmpa_bool
21 \bool_new:N \l_@@_disable_defaults_bool
22 \bool_new:N \l_@@_alias_bool
23 \bool_new:N \l_@@_external_bool
24 \bool_new:N \l_@@_defining_encoding_bool
25 \bool_new:N \l_@@_scriptlang_exist_bool
26 \bool_new:N \g_@@_em_normalise_slant_bool
27 \bool_new:N \l_@@_proceed_bool
```


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

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

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

Counters

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

Floats

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

Dimensions

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

Sequences

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

Comma-lists

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

```

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

```

Property lists

```

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

```

Token lists

Visible (perhaps?)

```

77 \tl_new:N \l_fontspec_family_tl
78 \tl_new:N \g_fontspec_encoding_tl
79 \tl_new:N \l_fontspec_fontname_tl

```

2e interactions

```

80 \tl_clear_new:N \UTFencname
81 \tl_clear_new:N \cyrillicencoding
82 \tl_clear_new:N \latinencoding

```

Renderer/shaper

```

83 \tl_new:N \l_@@_renderer_tl
84 \tl_new:N \l_@@_mode_tl
85 \tl_new:N \l_@@_shaper_tl

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

```

```

99 \tl_new:N \g_@@_nfss_enc_tl
100 \tl_new:N \g_@@_nfss_family_tl
101 \tl_new:N \l_@@_nfss_sc_tl
102 \tl_new:N \l_@@_nfss_tl
103 \tl_new:N \l_@@_nfss_fam_tl

104 \tl_new:N \l_@@_size_tl
105 \tl_new:N \l_@@_sizedfont_tl
106 \tl_new:N \l_@@_this_font_tl
107 \tl_new:N \l_@@_ttc_index_tl
108 \tl_new:N \l_@@_smcp_shape_tl

```

EM and STRONG

```

109 \tl_new:N \l_@@_emshape_query_tl
110 \tl_new:N \l_@@_em_switch_tl
111 \tl_new:N \l_@@_strong_switch_tl

```

Scratch variables

```

112 \tl_new:N \l_@@_tmp_tl
113 \tl_new:N \l_@@_tmpa_tl
114 \tl_new:N \l_@@_tmpb_tl
115 \tl_new:N \l_@@_em_tmp_tl
116 \tl_new:N \l_@@_strong_tmp_tl

```

Maths fonts

```

117 \tl_new:N \g_@@_mathrm_tl
118 \tl_new:N \g_@@_bfmathrm_tl
119 \tl_new:N \g_@@_mathsf_tl
120 \tl_new:N \g_@@_mathtt_tl

```

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

```

121 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
122 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
123 \tl_gset:Nn \g_@@_mathtt_tl {\ttdefault}

124 \tl_new:N \l_@@_family_label_tl
125 \tl_new:N \l_@@_fake_slant_tl
126 \tl_new:N \l_@@_fake_embolden_tl

```

Internal font names

```

127 \tl_new:N \l_@@_fontname_up_tl
128 \tl_new:N \l_@@_fontname_bf_tl
129 \tl_new:N \l_@@_fontname_it_tl
130 \tl_new:N \l_@@_fontname_bfit_tl
131 \tl_new:N \l_@@_fontname_sl_tl
132 \tl_new:N \l_@@_fontname_bfsl_tl
133 \tl_new:N \l_@@_fontname_sc_tl

```

Script and Language

```
134 \tl_new:N \l_@@_script_tl
135 \tl_new:N \l_@@_script_name_tl
136 \tl_set:Nn \l_@@_script_name_tl {CustomDefault}

137 \tl_new:N \l_@@_lang_tl
138 \tl_new:N \l_@@_lang_name_tl
139 \tl_set:Nn \l_@@_lang_name_tl {Default}
```

Generic font features

```
140 \tl_new:N \l_@@_scale_tl
141 \tl_new:N \l_@@_hyphenchar_tl
142 \tl_new:N \l_@@_hexcol_tl
143 \tl_new:N \l_@@_opacity_tl
144 \tl_new:N \l_@@_optical_size_tl
145 \tl_new:N \l_@@_mapping_tl
146 \tl_new:N \l_@@_punctspace_adjust_tl
147 \tl_new:N \l_@@_wordspace_adjust_tl
148 \tl_new:N \l_@@_postadjust_tl

149 \tl_const:Nn \c_@@_hexcol_tl {0000000}
150 \tl_const:Nn \c_@@_opacity_tl {FF~}
151 \tl_const:Nn \c_@@_postadjust_tl { \l_@@_wordspace_adjust_tl \l_@@_punctspace_adjust_tl }
```

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

```
152 \tl_new:N \g_@@_rawfeatures_sclist
153 \tl_new:N \l_@@_pre_feat_sclist
```

Font families

```
154 \tl_new:N \l_@@_rmfamily_family_tl
155 \tl_new:N \l_@@_sffamily_family_tl
156 \tl_new:N \l_@@_ttfamily_family_tl
```

File IV

fontspec-code-msg.dtx

1 Error/warning/info messages

Shorthands for messages:

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

Allow messages to be written with spaces acting as normal:

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

1.1 Errors

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

```

39   The feature #1 doesn't appear to be defined.
40   }
41   {
42   It looks like you're trying to rename a feature that doesn't exist.
43   }
44   \@@_msg_new:nnn {fontspec} {no-glyph}
45   {
46   '#1' does not contain glyph #2.
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} {ot-tag-too-long}
109    {
110      OpenType tag '#1' is too long; script, language, and feature tags must be four characters or
111    }
112    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist}
113    {
114      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
115      for AAT font '\l_fontspec_fontname_tl'.
116    }
117    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist-in-font}
118    {
119      AAT feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
120      in font '\l_fontspec_fontname_tl'.
121    }
122    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist}
123    {
124      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
125      for OpenType font '\l_fontspec_fontname_tl'
126    }
127    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist-in-font}
128    {
129      OpenType feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
130      for font '\l_fontspec_fontname_tl'
131      with script '\l_@@_script_name_tl' and language '\l_@@_lang_name_tl'.
132    }
133    \@@_msg_new:nnn {fontspec} {no-opticals}
134    {
135      '#1' doesn't appear to have an Optical Size axis.
136    }
137    \@@_msg_new:nnn {fontspec} {language-not-exist}
138    {

```

```

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

```

1.3 Info messages

```

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

```



```

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

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

```

File V

fontspec-code-opening.dtx

1 Opening code

1.1 Package options

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

1.2 Encodings

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

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

```

39 \bool_if:NTF \g_@@_euenc_bool
40 {
41   \tl_gset:Nn \g_fontspec_encoding_tl {EU1}
42   \tl_gset:Nn \g_fontspec_encoding_tl {EU2}
43 }
44 { \tl_gset:Nn \g_fontspec_encoding_tl { TU } }

45 \tl_set:Nn \rmdefault {lmr}
46 \tl_set:Nn \sfdefault {lmss}
47 \tl_set:Nn \ttdefault {lmtt}
48 \RequirePackage[\g_fontspec_encoding_tl]{fontenc}
49 \tl_set_eq:NN \UTFencname \g_fontspec_encoding_tl % for xunicode if needed

```

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

```

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

```

Dealing with a couple of the problems introduced by babel:

```

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

```

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

```

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

```

1.3 Generic functions

`\FontspecSetCheckBoolTrue`
`\FontspecSetCheckBoolFalse`

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

```

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

```

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

`\@@_keys_set_known:nnN`

```

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

```

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

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

```

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

```

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

\@@_lua_function:ne

\@@_lua_function:nee 78 {*LU}

\@@_lua_function:neee 79 \cs_set:Npn \@@_lua_function:ne #1#2 { \lua_now:e { fontspec.#1 ("#2")

\@@_lua_function:neeee 80 \cs_set:Npn \@@_lua_function:nee #1#2#3 { \lua_now:e { fontspec.#1 ("#2", "#3")

81 \cs_set:Npn \@@_lua_function:neee #1#2#3#4 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4")

82 \cs_set:Npn \@@_lua_function:neeee #1#2#3#4#5 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4", "#5")

83 {/LU}

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

1.4 expl3 variants

```

84 \cs_generate_variant:Nn \int_set:Nn {Nv}
85 \cs_generate_variant:Nn \keys_set:nn {nx}
86 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
87 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
88 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
89 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
90 \cs_generate_variant:Nn \prop_gput:Nnn {NxN}
91 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
92 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
93 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
94 \cs_generate_variant:Nn \tl_if_empty_p:n {e}
95 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
96 \cs_generate_variant:Nn \tl_if_empty:nF {x}
97 \cs_generate_variant:Nn \tl_if_empty:nF {f}
98 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
99 \cs_generate_variant:Nn \tl_replace_all:Nnn {Nnx}

```

File VI

fontspec-code-fontload.dtx

1 expl3 interface for primitive font loading

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

(End definition for \@@_primitive_font_set:Nnn and \@@_primitive_font_gset:Nnn. These functions are doc-
umented on page ??.)

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

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

\@@_primitive_font_if_null_p:N
\@@_primitive_font_if_null:NNTF
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:NNTF. This function is documented on page ??.)

\@@_primitive_font_set_p:NnnTF
\@@_primitive_font_set:NnnTFTF
21 \prg_set_conditional:Nnn \@@_primitive_font_set:Nnn {TF,T,F}
22 {
23   \@@_primitive_font_set:Nnn #1 {#2} {#3}
24   \@@_primitive_font_if_null:NNTF #1 {\prg_return_false:} {\prg_return_true:}
25 }
26 \prg_set_conditional:Nnn \@@_primitive_font_gset:Nnn {TF,T,F}
27 {
28   \@@_primitive_font_gset:Nnn #1 {#2} {#3}
29   \@@_primitive_font_if_null:NNTF #1 {\prg_return_false:} {\prg_return_true:}
30 }
31 \cs_set:Npn \@@_primitive_font_set:Omn { \exp_last_unbraced:No \@@_primitive_font_set:Nnn }
```

```

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

```

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

\@@_primitive_font_if_exist:nTF

```

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

```

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

\@@_primitive_font_glyph_if_exist:NnTF

```

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

```

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

\@@_primitive_font_set_hyphenchar:Nn

```

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

```

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

\@@_primitive_font_get_name:N

\@@_primitive_font_current_name:

```

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

```

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

File VII

fontspec-code-interfaces.dtx

1 User commands

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

```
1 \NewDocumentCommand \fontspec { 0{} m 0{} }
2 {
3   \@@_main_fontspec:nn {#1,#3} {#2}
4   \ignorespaces
5 }
6
7 \NewDocumentCommand \setmainfont { 0{} m 0{} }
8 {
9   \@@_main_setmainfont:nn {#1,#3} {#2}
10  \ignorespaces
11 }
12
13 \NewDocumentCommand \setsansfont { 0{} m 0{} }
14 {
15   \@@_main_setsansfont:nn {#1,#3} {#2}
16   \ignorespaces
17 }
18
19 \NewDocumentCommand \setmonofont { 0{} m 0{} }
20 {
21   \@@_main_setmonofont:nn {#1,#3} {#2}
22   \ignorespaces
23 }
24
25 \NewDocumentCommand \setmathrm { 0{} m 0{} }
26 {
27   \@@_main_setmathrm:nn {#1,#3} {#2}
28 }
29
30 \NewDocumentCommand \setboldmathrm { 0{} m 0{} }
31 {
32   \@@_main_setboldmathrm:nn {#1,#3} {#2}
33 }
34
35 \NewDocumentCommand \setmathsf { 0{} m 0{} }
36 {
37   \@@_main_setmathsf:nn {#1,#3} {#2}
38 }
39
40 \NewDocumentCommand \setmathtt { 0{} m 0{} }
41 {
42   \@@_main_setmathtt:nn {#1,#3} {#2}
43 }
```

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

```

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

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

41 \NewDocumentCommand \newfontfamily { m O{} m O{} }
42 {
43     \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
44 }

45 \NewDocumentCommand \renewfontfamily { m O{} m O{} }
46 {
47     \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
48 }

49 \NewDocumentCommand \setfontfamily { m O{} m O{} }
50 {
51     \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
52 }

53 \NewDocumentCommand \providefontfamily { m O{} m O{} }
54 {
55     \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
56 }

57 \NewDocumentCommand \newfontface { m O{} m O{} }
58 {
59     \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
60 }

61 \NewDocumentCommand \renewfontface { m O{} m O{} }
62 {
63     \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
64 }

65 \NewDocumentCommand \setfontface { m O{} m O{} }
66 {
67     \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
68 }

69 \NewDocumentCommand \providefontface { m O{} m O{} }
70 {
71     \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
72 }

```

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

```

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

```



```

78     \ignorespaces
79   }

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

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

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

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

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

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

```

`\newICUfeature` Deprecated.

```

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

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

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

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

`\liningnums`

```

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

```

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

File VIII

fontspec-code-user.dtx

1 User command internals

1.1 Font selection

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

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

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

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

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

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

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

`\setsansfont` Same as above.

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

```

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

```

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

`\setmonofont` Same as above.

```

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

```

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

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

```

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

```

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

`\setboldmathrm`

```

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

```

```

69 (XE) \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {#1} {#2}
70 (LU) \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {Renderer=Basic,#1} {#2}
71     \@@_setboldmathrm_hook:nn {#1} {#2}
72 }

```

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

`\setmathsf`

```

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

```

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

`\setmathtt`

```

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

```

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

Hooks:

```

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

```

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

```

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

```

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

```

96 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
97 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
98 \tl_gset:Nn \g_@@_mathtt_tl {\ttdefault}

```

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

```

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

```

```

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

```

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

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

```

112 \cs_new:Nn \@@_main_newfontface:NnnN
113 {
114     \@@_main_newfontfamily:NnnN #1 { BoldFont={},ItalicFont={},SmallCapsFont={},#2 } {#3} #4
115 }

```

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

1.2 Font feature selection

\@@_set_default_features:nn

```

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

```

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

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

```

121 \cs_new:Nn \@@_set_font_default_features:nnn
122 {
123     <debug> \typeout{\unexpanded{_set_font_default_features:nnn:{#1}{#2}{#3}}}
124     \clist_map_inline:nn {#2}
125     {
126         \tl_if_single:nTF {##1}
127         { \tl_set:Nn \l_@@_tmp_tl { \cs:w l_@@_ \cs_to_str:N ##1 _family_tl\cs_end: } }
128         { \@@_sanitise_fontname:Nn \l_@@_tmp_tl {##1} }
129
130         \IfBooleanTF {#1}
131         {
132             \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
133             { \tl_clear:N \l_@@_tmpb_tl }
134             \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
135             \prop_gput:NVV \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
136         }
137     }

```

```

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

```

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

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

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

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

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

```

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

```

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

1.3 Defining new font features

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

`\aliasfontfeatureoption`

```

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

```

```

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

```

```
\@@ main DeclareFontExtensions:n
```

Defaults:

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

1.4 High level conditionals

```

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

```

1.5 \oldstylenums and \liningnums

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

```
\liningnums 285 \cs_new_protected:Nn \@@_main_oldstylenums:n
286 {
287   \group_begin:
288     \addfontfeature{Numbers=OldStyle}
289     #1
290   \group_end:
291 }
292 \cs_new_protected:Nn \@@_main_liningnums:n
293 {
294   \group_begin:
295     \addfontfeature{Numbers=Lining}
296     #1
297   \group_end:
298 }
```

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

File IX

fontspec-code-api.dtx

1 Programmer's interface

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

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

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

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

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

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

144     { \prg_return_false: }
145 }

```

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

`\fontspec_font_if_exist:n`

```

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

```



```

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

```

```

180 \cs_set_eq:NN \IfFontExistsTF \fontspec_font_if_exist:nTF

```

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

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

```

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

```

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

`\fontspec_if_small_caps:TF`

```

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

```

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

File X

fontspec-code-internal.dtx

1 Internals

1.1 The main function for setting fonts

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

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

Some often-used variables to know about:

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

\@@_load_external_fontoptions:Nn Load a possible .fontspec font configuration file. This file could set font-specific options for the font about to be loaded.

```

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

```

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

\@@_extract_all_features:

```

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

```

```

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

```

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

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

Perform the (multi-step) feature parsing process.

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

```

118 \cs_new:Nn \@@_preparse_features:
119 {
120   <debug> \typeout{:: @@_preparse_features:}

```

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

```

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

```

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

```

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

```

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

\@@_load_font:

```

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

```

```

134 <debug>\typeout{:: @@_load_font}
135
136 <debug>\typeout{Set~ base~ font~ for~ preliminary~ analysis: \@@_construct_font_call:nn { \l_@
137 \@@_primitive_font_set:NnnF \l_@@_test_font
138 { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} }
139 { \f@size pt - 2sp }
140 { \@@_error:nx {font-not-found} {\l_@@_fontname_up_tl} }
141
142 <debug>\typeout{Set~ base~ font~ properly: \@@_construct_font_call:nn { \l_@@_fontname_up_tl }
143 \@@_set_font_type:N \l_@@_test_font
144 \@@_primitive_font_gset:0nn \l_@@_fontface_cs_tl
145 { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} }
146 { \f@size pt }
147
148 \l_@@_fontface_cs_tl % this is necessary for LuaLaTeX to check the scripts properly
149
150 }

```

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

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

#2 : Extension

#3 : TTC Index

#4 : Renderer

#5 : Optical size

#6 : Font features

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

```

151 \cs_new:Nn \@@_construct_font_call:nnnnnn
152 {
153 <XE> " \@@_fontname_wrap:n { #1 #2 #3 }
154 <LU> " \@@_fontname_wrap:n { #1 #2 } #3
155 #4 #5
156 \str_if_eq:eeF {#6}{ } {:#6} "
157 }

```

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

```

158 \cs_new:Nn \@@_construct_font_call:nn
159 {
160 \@@_construct_font_call:nnnnnn
161 {#1}
162 \l_@@_extension_tl
163 \l_@@_ttc_index_tl
164 \l_@@_renderer_tl
165 \l_@@_optical_size_tl
166 {#2}
167 }

```

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

`\@@_font_is_file:` The `\@@_fontname_wrap:n` command takes the font name and either passes it through unchanged or wraps it in the syntax for loading a font ‘by filename’. Xe_{La}TeX’s syntax is followed since luaotfload provides compatibility.

```

168 \cs_new:Nn \@@_font_is_name:
169 {
170   \cs_set_eq:NN \@@_fontname_wrap:n \use:n
171 }

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

```

(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.

```

176 \cs_new:Nn \@@_set_scriptlang:
177 {
178   <debug> \typeout{:: _set_scriptlang:}
179   \bool_if:NT \l_@@_firsttime_bool
180   {
181     \tl_if_empty:NF \l_@@_script_name_tl
182     {
183       <debug> \typeout{::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
184       \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
185       \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
186     }
187   }
188 }

```

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

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

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

```

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

```

```

204         \bool_if:nT { \l_@@_atsui_bool || \l_@@_graphite_bool }
205         { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
206     }
207 </XE>
208 <*LU>
209 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features::~\l_@@_keys_leftover_clist
210         \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
211 </LU>
212
213     \tl_if_empty:NF \l_@@_mapping_tl
214     { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
215
216     \str_if_eq:eeF { \l_@@_hexcol_tl \l_@@_opacity_tl }
217     { \c_@@_hexcol_tl \c_@@_opacity_tl }
218     { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
219 }

```

(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.

```

220 \prg_new_conditional:Nnn \@@_save_family_needed:n { TF }
221 {
222
223 <debug> \typeout{save~ family::~ #1}
224 <debug> \typeout{== fontid_tl: "\l_@@_fontid_tl".}
225
226     \tl_if_empty:NTF \l_@@_nfss_fam_tl
227     {
228         \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
229         {
230             \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
231             \prg_return_false:
232         }
233         {
234             \tl_set:Nx \l_@@_tmp_tl {#1}
235             \tl_remove_all:Nn \l_@@_tmp_tl { ~ }
236             \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl
237             \prg_return_true:
238         }
239     }
240     {
241         \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
242         \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl _prop }
243         \prg_return_true:
244     }
245 }

```



```

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

```

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

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

```

259 \cs_new:Nn \@@_save_family:nn
260 {
261   \@@_save_fontinfo:n {#2}
262   \@@_find_autofonts:
263   \DeclareFontFamily{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{-}
264   \@@_set_faces:
265   \@@_info:nxx {defining-font} {#1} {#2}
266 }

```

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

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

```

267 \cs_new:Nn \@@_save_fontinfo:n
268 {
269   \prop_new:c {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop}
270   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
271   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options} { \l_@@_all_features }
272   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
273   {
274     \@@_construct_font_call:nn {\l_fontspeg_fontname_tl}
275     { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
276   }
277   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
278   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num} \l_@@_language_int
279   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_@@_script_tl
280   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_@@_lang_tl
281 }

```

(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:`

```

282 \cs_new:Nn \@@_find_autofonts:
283 {
284   \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
285   {
286     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
287     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
288     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspeg_fontname_tl} {/BI}
289   }
290
291   \bool_if:NF \l_@@_nobf_bool
292   {
293     \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspeg_fontname_tl} {/B}
294   }
295
296   \bool_if:NF \l_@@_noit_bool
297   {
298     \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspeg_fontname_tl} {/I}
299   }
300
301   \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
302 }

```

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

`\@@_set_faces:`

```

303 \cs_new:Nn \@@_set_faces:
304 {
305   \@@_add_nfssfont:nnnn \mddefault \updefault \l_fontspeg_fontname_tl \l_@@_fontfeat_up
306   \@@_add_nfssfont:nnnn \bfdefault \updefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_clist
307   \@@_add_nfssfont:nnnn \mddefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_clist
308   \@@_add_nfssfont:nnnn \mddefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_clist
309   \@@_add_nfssfont:nnnn \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit_cli
310   \@@_add_nfssfont:nnnn \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl_cli
311
312   \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nnnnn ##2 }
313 }
314 \cs_new:Nn \@@_set_faces_aux:nnnnn
315 {
316   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
317   \@@_make_font_shapes:Nnnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
318 }

```

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

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

```

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

```

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

```

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

```

```

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

```

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

1.2.1 Fonts

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

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

```

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

```

```

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

```

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

LuaTeX only supports one:

```

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

```

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

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

This function looks for font with $\langle name \rangle$ 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_YTeX 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.

```

373 \cs_new:Nn \@@_set_autofont:Nnn
374 {
375   \bool_if:NF \l_@@_external_bool
376   {
377     \tl_if_empty:xF {#2}
378     {
379       \tl_if_empty:NT #1
380       {
381         \@@_if_autofont:nnTF {#2} {#3}
382         { \tl_set:Nx #1 {#2#3} }
383         { \@@_info:nx {no-font-shape} {#2#3} }
384       }

```

```

385     }
386   }
387 }
388 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
389 {
390   \group_begin:
391   \@@_primitive_font_set:Nnn \l_@@_tmpa_font { \@@_construct_font_call:nn {#1} {} } { \f@size pt }
392   \@@_primitive_font_set:Nnn \l_@@_tmpb_font { \@@_construct_font_call:nn {#1#2} {} } { \f@size pt }
393   \str_if_eq:eeTF { \@@_primitive_font_get_name:N \l_@@_tmpa_font } { \@@_primitive_font_get_name:N \l_@@_tmpb_font }
394   { \group_end: \prg_return_false: }
395   { \group_end: \prg_return_true: }
396 }

```

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

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

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

```

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

```

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

\@@_declare_shape:nnnn #1 : Font series
 #2 : Font shape

#3 : Font features

#4 : Size features

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

The main part is to loop through SizeFeatures arguments, which are of the form

SizeFeatures={{<one>},{<two>},{<three>}}.

```
421 \cs_new:Nn \@@_declare_shape:nnnn
422 {
423   <debug>\typeout{~= declare_shape:~{\l_fontspeg_fontname_tl}~{#1}~{#2}}
424   \tl_clear:N \l_@@_nfss_tl
425   \tl_clear:N \l_@@_nfss_sc_tl
426   \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspeg_fontname_tl
427
428   \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
429
430   \@@_declare_shapes_normal:nn {#1} {#2}
431   \@@_declare_shapes_smcaps:nn {#1} {#2}
432   \@@_declare_shape_slanted:nn {#1} {#2}
433   \@@_declare_shape_loginfo:nn {#1} {#2}
434 }
```

```
435 \cs_generate_variant:Nn \@@_declare_shape:nnnn {nnxx}
```

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

\@@_setup_single_size:nn

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

```

460 <debug>\typeout{====~Small~ caps~ found.}
461         \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
462     }
463     {
464 <debug>\typeout{====~Small~ caps~ not~ found.}
465         \bool_set_true:N \l_@@_nosc_bool
466     }
467 }
468 { \@@_load_fontname:Nn \l_fontspec_fontname_tl {\l_@@_fontname_sc_tl} }% local for e
469 }
470
471 \bool_if:NF \l_@@_nosc_bool
472 {
473     \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
474     {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
475 }
476 }

```

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

\@@_setup_nfss:Nnnn

```

477 \cs_new:Nn \@@_setup_nfss:Nnnn
478 {
479 <debug>\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}
480
481     \@@_get_features:n { #2 , #3 , #4 }
482 <debug>\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist}
483
484     \tl_if_empty:NF \l_@@_scale_tl
485     {
486         \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
487     }
488
489     \tl_put_right:Nx #1
490     {
491         <\l_@@_size_tl> \l_@@_scale_tl
492         \@@_construct_font_call:nn { \l_fontspec_fontname_tl }
493         { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
494     }
495 }

```

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

\@@_declare_shapes_normal:nn

```

496 \cs_new:Nn \@@_declare_shapes_normal:nn
497 {
498     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
499     {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
500 }

```

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

`\@@_declare_shapes_smcaps:nn`

```

501 \cs_new:Nn \@@_declare_shapes_smcaps:nn
502 {
503   \tl_if_empty:NF \l_@@_nfss_sc_tl
504   {
505     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl} {#1}
506     { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
507   }
508 }
509 \cs_new:Nn \@@_combo_sc_shape:n
510 {
511   \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
512   { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
513   { \scdefault }
514 }

```

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

`\@@_DeclareFontShape:nnnnnn`

```

515 \cs_new:Nn \@@_DeclareFontShape:nnnnnn
516 {
517   <debug>\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
518   \group_begin:
519   \normalsize
520   \cs_undefine:c {#1/#2/#3/#4/\f@size}
521   \group_end:
522   \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
523   }
524   \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`

```

525 \cs_new:Nn \@@_declare_shape_slanted:nn
526 {
527   \bool_if:nT
528   {
529     \str_if_eq_p:ee {#2} {\itdefault} &&
530     !(\str_if_eq_p:ee {\itdefault} {\sldefault})
531   }
532   {
533     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}
534     {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
535   }
536 }

```


Lastly some informative messaging.

```

\@@_declare_shape_loginfo:nn 537 \cs_new:Nn \@@_declare_shape_loginfo:nn
538 {
539   \tl_gput_right:Nx \g_@@_defined_shapes_tl
540   {
541     \exp_not:n { \ }
542     ~ \exp_not:N \str_case:nn {#1/#2}
543     {
544       {\mddefault/\updefault} {'normal'~}
545       {\bfdefault/\updefault} {'bold'~}
546       {\mddefault/\itdefault} {'italic'~}
547       {\mddefault/\sldefault} {'slanted'~}
548       {\bfdefault/\itdefault} {'bold~ italic'~}
549       {\bfdefault/\sldefault} {'bold~ slanted'~}
550     } (#1/#2)~
551     with~ NFSS~ spec.:~
552     \l_@@_nfss_tl
553     \exp_not:n { \ }
554     ~ \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
555     {
556       {\mddefault/\scdefault} {'small~ caps'~}
557       {\bfdefault/\scdefault} {'bold~ small~ caps'~}
558       {\mddefault/\itscdefault} {'italic~ small~ caps'~}
559       {\bfdefault/\itscdefault} {'bold~ italic~ small~ caps'~}
560       {\mddefault/\slscdefault} {'slanted~ small~ caps'~}
561       {\bfdefault/\slscdefault} {'bold~ slanted~ small~ caps'~}
562     }~( #1 / \@@_combo_sc_shape:n {#2} )~
563     with~ NFSS~ spec.:~
564     \l_@@_nfss_sc_tl
565     \tl_if_empty:IF {\l_@@_postadjust_tl}
566     {
567       \exp_not:N \ and~ font~ adjustment~ code:
568       \exp_not:N \ \l_@@_postadjust_tl
569     }
570   }
571 }

```

Maybe \str_if_eq:eeF would be better?

1.2.2 Features

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

```

\l_@@_pre_feat_sclist 572 \tl_set:Nn \l_@@_pre_feat_sclist
573 {*XE}
574 {
575   \bool_if:NT \l_@@_ot_bool
576   {
577     \tl_if_empty:NF \l_@@_script_tl
578     {
579       script = \l_@@_script_tl ;
580       language = \l_@@_lang_tl ;
581     }

```

```

582     }
583   }
584   </XE>
585   <*LU>
586   {
587     mode      = \l_@@_mode_tl    ;
588     \tl_if_empty:NF \l_@@_shaper_tl
589     {
590       shaper = \l_@@_shaper_tl    ;
591     }
592     \tl_if_empty:NF \l_@@_script_tl
593     {
594       script  = \l_@@_script_tl ;
595       language = \l_@@_lang_tl  ;
596     }
597   }
598   </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF
599   <LU>\cs_new:Nn \@@_make_smallcaps:TF
600   <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
601   {
602     \exp_args:No \@@_check_ot_feat:NnTF \l_@@_fontface_cs_tl {smcp} {#1} {#2}
603   }
604   <*XE>
605   \cs_new:Nn \@@_make_smallcaps:TF
606   {
607     \bool_if:NTF \l_@@_ot_bool
608     { \@@_make_ot_smallcaps:TF {#1} {#2} }
609     {
610       \bool_if:NT \l_@@_atsui_bool
611       {
612         \exp_args:No \@@_make_AAT_feature_string:NnnTF
613         \l_@@_fontface_cs_tl {3} {3} {#1} {#2}
614       }
615     }
616   }
617   </XE>

```

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

```

\@@_update_featstr:n
618   \cs_new:Nn \@@_update_featstr:n
619   {
620     <debug> \typeout{::: @@_update_featstr:n {#1}}
621     \bool_if:NF \l_@@_firsttime_bool
622     {
623       \tl_gset:Nx \g_@@_single_feat_tl { #1 }
624     <debug> \typeout{:::~ Adding~ feature.}
625     \tl_gput_right:Nx \g_@@_rawfeatures_sclist {#1;}
626   }
627   }

```

```

\@@_remove_clashing_featstr:n 628 \cs_new:Nn \@@_remove_clashing_featstr:n
629 {
630 (debug) \typeout{::: @@_remove_clashing_featstr:n {#1}}
631 \clist_map_inline:nn {#1}
632 {
633 (debug) \typeout{:::~ Removing~ feature~ "##1;}
634 \tl_gremove_all:Nn \g_@@_rawfeatures_sclist {##1;}
635 }
636 }
637 \cs_generate_variant:Nn \@@_remove_clashing_featstr:n {x}

```

1.3 Initialisation

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

```

\@@_init: 638 \cs_set:Npn \@@_init:
639 {
640 (debug) \typeout{:: @@_init:}
641 \bool_set_false:N \l_@@_ot_bool
642 \bool_set_true:N \l_@@_firsttime_bool
643 \@@_font_is_name:
644 \tl_clear:N \l_@@_font_path_tl
645 \tl_clear:N \l_@@_optical_size_tl
646 \tl_clear:N \l_@@_ttc_index_tl
647 \tl_clear:N \l_@@_renderer_tl
648 \tl_gclear:N \g_@@_defined_shapes_tl
649 \tl_gclear:N \g_@@_curr_series_tl
650 \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspeg_encoding_tl
651 (*LU)
652 \tl_set:Nn \l_@@_mode_tl {node}
653 \int_set:Nn \prehyphenchar { \- } % fixme
654 \int_zero:N \posthyphenchar % fixme
655 \int_zero:N \preexhyphenchar % fixme
656 \int_zero:N \postexhyphenchar % fixme
657 </LU>
658 }

```

Executed in \@@_get_features:Nn.

```

\@@_init_fontface: 659 \cs_new:Nn \@@_init_fontface:
660 {
661 \tl_gclear:N \g_@@_rawfeatures_sclist
662 \tl_clear:N \l_@@_scale_tl
663 \tl_set_eq:NN \l_@@_opacity_tl \c_@@_opacity_tl
664 \tl_set_eq:NN \l_@@_hexcol_tl \c_@@_hexcol_tl
665 \tl_set_eq:NN \l_@@_postadjust_tl \c_@@_postadjust_tl
666 \tl_clear:N \l_@@_wordspace_adjust_tl
667 \tl_clear:N \l_@@_punctspace_adjust_tl
668 }

```

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

```
\@@_ot_validate_tag:n 669 <*LU>
670 \cs_new_protected:Nn \@@_ot_validate_tag:n
671 {
672   \@@_ot_validate_tag:w #1 \q_nil
673 }
674 \cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
675 \cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
676 {
677   \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
678     { \@@_ot_validate_tag_aux:w #2 \c_empty_tl \c_empty_tl \q_nil }
679     { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
680 }
681 \cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
682 {
683   \int_compare:nT { \tl_count:n {#5} > 2 }
684     { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
685 }
686 </LU>
```

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

\@@_iv_str_to_num:Nn

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

```
687 <*XE>
688 \cs_new:Nn \@@_iv_str_to_num:Nn
689 {
690   (debug)\typeout{iv_str_to_num:~#1~/~#2}
691   \@@_strip_leading_sign:Nw #1#2 \q_nil
692 }
693 \cs_generate_variant:Nn \@@_iv_str_to_num:Nn {Nx}
```

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

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

```
694 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
695 {
696   \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
697     { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
698     { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
699 }
```

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

```
700 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
```

```

701 {
702   \int_compare:nT { \tl_count:n {#6} > 2 }
703     { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }
704
705   \int_set:Nn #1
706     {
707       `#2 * "10000000
708       + `#3 * "10000
709       + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
710       + \ifx \c_empty_tl #5 32 \else `#5 \fi
711     }
712   }
713 </XE>

```

```

\@@_lang_dflt_correct:N 714 <*XE>
715 \cs_new_protected:Nn \@@_lang_dflt_correct:N
716 {
717   \int_compare:nNnT {#1} = {1145457748} % "DFLT"
718     {
719       \int_zero:N #1
720     }
721   }
722 </XE>

```

File XI

fontspec-code-opentype.dtx

1 OpenType definitions code

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

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

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

```

37         }
38     }

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

39 \cs_new:Nn \@@_feat_off:n {#10ff}
40 \cs_new:Nn \@@_feat_reset:n {#1Reset}

41 \cs_new:Nn \@@_define_opentype_onoffreset:nnnnn
42 {
43     \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} {#2} {#3} {+#4} {#5}
44     \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_off:n {#2} } {#3} {-#4}
45     \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {+#4}
46 }

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

47 \cs_new:Nn \@@_define_opentype_onreset:nnnnn
48 {
49     \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} {#2} {#3} {#4} {#5}
50     \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {#4}
51 }

```

1.1 Adding features when loading fonts

When remove clashing features,

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

```

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

```

```

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

```

(End definition for \@@_DeclareFontShape: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. \l_@@_script_int is used to store the number corresponding to the script tag string.

```

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

```



```

111       \prg_return_true:
112     }
113     {
114   <debug>\typeout{:::~ FALSE}
115       \prg_return_false:
116     }
117   </LU>
118 }
119 }
120 }

```

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

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

```

121 \prg_new_conditional:Nnn \@@_check_lang:Nn {TF}
122 {
123   \@@_check_lang:NnnTF #1 {#2} {\l_@@_script_tl} {\prg_return_true:} {\prg_return_false:}
124 }
125 \prg_new_conditional:Nnn \@@_check_lang:Nnn {TF}
126 {
127   <debug>\typeout{:: _check_lang:Nn~#1~/~#2~/~#3~/}
128   \bool_if:NTF \l_@@_never_check_bool
129     { \prg_return_true: }
130     {
131       \bool_if:nTF { \tl_if_empty_p:e {#3} }
132         { \prg_return_false: }
133         {
134       (*XE)
135         \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
136         \@@_iv_str_to_num:Nx \l_@@_script_int {#3}
137         \int_set:Nn \l_@@_tmpb_int
138           { \XeTeXOTcountlanguages #1 \l_@@_script_int }
139         \int_zero:N \l_@@_tmpa_int
140         \bool_set_false:N \l_@@_fontspec_check_bool
141         \bool_until_do:nn { \int_compare_p:nNn \l_@@_tmpa_int = \l_@@_tmpb_int }
142           {
143             \int_set:Nn \l_@@_tmpc_int
144               { \XeTeXOTlanguagetag #1 \l_@@_script_int \l_@@_tmpa_int }
145
146             \int_compare:nNnTF \l_@@_tmpc_int = \l_@@_strnum_int
147               {
148                 \bool_set_true:N \l_@@_fontspec_check_bool
149                 \int_set:Nn \l_@@_tmpa_int {\l_@@_tmpb_int}
150               }
151               {
152                 \int_incr:N \l_@@_tmpa_int
153               }
154           }

```

```

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

```

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

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

```

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

```

```

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

```

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

1.2 OpenType feature information

```

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

```

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

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

TODO: move the above elsewhere!!

File XII

fontspec-code-graphite.dtx

1 Graphite/AAT code

`\@@_define_aat_feature_group:n`

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

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

`\@@_define_aat_feature:nnnn`

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

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

`\@@_make_AAT_feature:nn`

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

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

`\@@_make_AAT_feature_string: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 58).

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_YTeX doesn't return a selector string for this number, since the feature is defined for the 'switching on' value. So we need to

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

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

```

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

```

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

File XIII

fontspec-code-keyval.dtx

1 Font loading (keyval) definitions

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

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

Wrapper function to save some characters in the source:

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

For catching features that cannot be used in \addfontfeatures:

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

1.1 Pre-pre-parsing stages

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

Don't load font config file

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

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

```
24 \@@_keys_define_code:nnn {fontspec-preparse-external} {Path}
25 {
26   \bool_set_true:N \l_@@_nobf_bool
27   \bool_set_true:N \l_@@_noit_bool
28   \bool_set_true:N \l_@@_external_bool
29 }
```



```

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

```

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

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

```

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

```

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

```

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

```

```

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

```

1.2 Pre-parsed features

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

```

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

```

Exactly the same:

```

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

```

TTC font index

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

1.3 Font faces

Upright

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

Italic and slanted

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

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

```
145 %\@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSeries}
146 % {
147 %   \tl_gset:Nx \g_@@_curr_series_tl { #1 }
148 %   \seq_put_right:Nx \l_@@_bf_series_seq { #1 }
149 % }
```

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

```

150 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldFont}
151 {
152   \tl_if_empty:nTF {#1}
153   {
154     \bool_set_true:N \l_@@_nobf_bool
155   }
156   {
157     \bool_set_false:N \l_@@_nobf_bool
158     \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
159
160     \seq_if_empty:NT \l_@@_bf_series_seq
161     {
162       \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
163       \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
164     }
165
166     \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
167     {
168       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
169     }
170
171     \prop_put:NxV \l_@@_nfss_prop {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
172
173     <debug>\typeout{Setting-bold-font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
174
175   }
176 }

```

Bold italic/slanted

```

177 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldItalicFont}
178 {
179   \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
180 }
181 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSlantedFont}
182 {
183   \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
184 }

```

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

```

185 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
186 {
187   \tl_if_empty:nTF {#1}
188   {
189     \bool_set_true:N \l_@@_nosc_bool
190   }
191   {
192     \bool_set_false:N \l_@@_nosc_bool
193     \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
194   }

```

```
195 }
```

1.3.1 Prepared font features

```
196 \@@_keys_define_code:nnn {fontspec-preparse} {UprightFeatures}
197 {
198   \clist_set:Nn \l_@@_fontfeat_up_clist {#1}
199 }
200 \@@_keys_define_code:nnn {fontspec-preparse} {BoldFeatures}
201 {
202   \clist_set:Nn \l_@@_fontfeat_bf_clist {#1}
203 }
204 % \prop_put:NxV \l_@@_nfss_prop
205 %   {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
206 }
207 \@@_keys_define_code:nnn {fontspec-preparse} {ItalicFeatures}
208 {
209   \clist_set:Nn \l_@@_fontfeat_it_clist {#1}
210 }
211 \@@_keys_define_code:nnn {fontspec-preparse} {BoldItalicFeatures}
212 {
213   \clist_set:Nn \l_@@_fontfeat_bfit_clist {#1}
214 }
215 \@@_keys_define_code:nnn {fontspec-preparse} {SlantedFeatures}
216 {
217   \clist_set:Nn \l_@@_fontfeat_sl_clist {#1}
218 }
219 \@@_keys_define_code:nnn {fontspec-preparse} {BoldSlantedFeatures}
220 {
221   \clist_set:Nn \l_@@_fontfeat_bfsl_clist {#1}
222 }
```

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

```
223 \@@_keys_define_code:nnn {fontspec} {SmallCapsFeatures}
224 {
225   \bool_if:NF \l_@@_firsttime_bool
226   {
227     \clist_set:Nn \l_@@_fontfeat_sc_clist {#1}
228   }
229 }
```

Features varying by size

```
230 \@@_keys_define_code:nnn {fontspec-preparse} {SizeFeatures}
231 {
232   \clist_set:Nn \l_@@_sizefeat_clist {#1}
233   \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
234 }
235 \@@_keys_define_code:nnn {fontspec-preparse-nested} {SizeFeatures}
236 {
237   \clist_set:Nn \l_@@_sizefeat_clist {#1}
238   \tl_if_empty:NT \l_@@_this_font_tl
239   { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
240 }
```

```

241 \@@_keys_define_code:nnn {fontspec-preparse-nested} {Font}
242 {
243   \tl_set:Nn \l_@@_this_font_tl {#1}
244 }
245 \@@_keys_define_code:nnn {fontspec} {SizeFeatures}
246 {
247   % dummy
248 }
249 \@@_keys_define_code:nnn {fontspec} {Font}
250 {
251   % dummy
252 }
253 \@@_keys_define_code:nnn {fontspec-sizing} {Size}
254 {
255   \tl_set:Nn \l_@@_size_tl {#1}
256 }
257 \@@_keys_define_code:nnn {fontspec-sizing} {Font}
258 {
259   \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
260 }

```

1.4 General font-independent features

These features can be applied to any font.

NFSS encoding For the very brave.

```

261 \@@_keys_define_code:nnn {fontspec-preparse} {NFSSEncoding}
262 {
263   \tl_gset:Nx \g_@@_nfss_enc_tl { #1 }
264 }

```

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.)

```

265 \@@_keys_define_code:nnn {fontspec-preparse} {NFSSFamily}
266 {
267   \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
268 }

```

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

```

269 \@@_keys_define_code:nnn {fontspec} {FontFace}
270 {
271   \tl_clear:N \l_@@_this_font_tl
272   \clist_set:No \l_@@_arg_clist { \use_iii:nnn #1 }
273   \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
274   \int_compare:nT { \clist_count:N \l_@@_arg_clist = 1 }
275   {
276     <debug>\typeout{FontFace~ parsing:~ one~ clist~ item}
277     \tl_if_in:NnF \l_@@_arg_clist {=}
278     {

```

```

279 <debug>\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
280     \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
281     \tl_clear:N \l_@@_this_feat_clist
282   }
283 }
284
285 \@@_add_nfssfont:nnnn
286   {\use_i:nnn #1} {\use_ii:nnn #1} {\l_@@_this_font_tl} {\l_@@_this_feat_clist}
287 }

```

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.

```

288 \@@_keys_define_code:nnn {fontspec} {Scale}
289 {
290   \str_case:nnF {#1}
291   {
292     {MatchLowercase} { \@@_calc_scale:n {5} }
293     {MatchUppercase} { \@@_calc_scale:n {8} }
294   }
295   { \tl_set:Nx \l_@@_scale_tl {#1} }
296 }

```

ScaleAgain

```

297 \@@_keys_define_code:nnn {fontspec} {ScaleAgain}
298 {
299   \tl_if_empty:NT \l_@@_scale_tl { \tl_set:Nn \l_@@_scale_tl {1} }
300   \tl_set:Nx \l_@@_scale_tl { \fp_eval:n { #1 * \l_@@_scale_tl } }
301   \@@_info:n {set-scale}
302 }

```

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

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

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

```

303 \cs_new:Nn \@@_calc_scale:n
304 {
305   \group_begin:
306
307   \fontencoding { \encodingdefault }
308   \fontfamily { \rmdefault }
309   \selectfont
310
311   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font

```

```

312 \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_@@_fontface_cs_tl
313
314 \tl_set:Nx \l_@@_scale_tl
315 {
316   \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
317               \dim_to_fp:n {\l_@@_tmpb_dim} }
318 }
319
320 \@@_info:n {set-scale}
321 \exp_args:NNNx
322 \group_end:
323 \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
324 }

```

(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’.

```

325 \cs_new:Nn \@@_set_font_dimen:NnN
326 {
327   \dim_set:Nn #1 { \fontdimen #2 #3 }
328   \dim_compare:nNnT #1 = {0pt}
329   {
330     \settoheight #1
331     {
332       \str_if_eq:nnTF {#3} {\font} \rmfamily #3
333       \int_case:nnF #2
334       {
335         {5} {x} % x-height
336         {8} {X} % cap-height
337         {?} % "else" clause; never reached.
338       }
339     }
340   }

```

(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.

```

341 \@@_keys_define_code:nnn {fontspec} {WordSpace}
342 {
343   \bool_if:NF \l_@@_firsttime_bool
344   { \_fontspec_parse_wordspace:w #1,,,\q_stop }
345 }
346 \@@_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}.

```

347 \cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
348 {

```



```

349 \tl_if_empty:nTF {#4}
350 {
351   \tl_set:Nn \l_@@_wordspace_adjust_tl
352   {
353     \fontdimen 2 \font = #1 \fontdimen 2 \font
354     \fontdimen 3 \font = #1 \fontdimen 3 \font
355     \fontdimen 4 \font = #1 \fontdimen 4 \font
356   }
357 }
358 {
359   \tl_set:Nn \l_@@_wordspace_adjust_tl
360   {
361     \fontdimen 2 \font = #1 \fontdimen 2 \font
362     \fontdimen 3 \font = #2 \fontdimen 3 \font
363     \fontdimen 4 \font = #3 \fontdimen 4 \font
364   }
365 }
366 }

```

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

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

```

367 \@@_keys_define_code:nnn {fontspec} {PunctuationSpace}
368 {
369   \str_case:e:nnF {#1}
370   {
371     {WordSpace}
372     {
373       \tl_set:Nn \l_@@_punctspace_adjust_tl
374       { \fontdimen 7 \font = 0 \fontdimen 2 \font }
375     }
376     {TwiceWordSpace}
377     {
378       \tl_set:Nn \l_@@_punctspace_adjust_tl
379       { \fontdimen 7 \font = 1 \fontdimen 2 \font }
380     }
381   }
382   {
383     \tl_set:Nn \l_@@_punctspace_adjust_tl
384     { \fontdimen 7 \font = #1 \fontdimen 7 \font }
385   }
386 }
387 \@@_aff_error:n {PunctuationSpace}

```

Secret hook into the font-adjustment code

```

388 \@@_keys_define_code:nnn {fontspec} {FontAdjustment}
389 {
390   \tl_put_right:Nx \l_@@_postadjust_tl {#1}
391 }

```

Letterspacing

```

392 \@@_keys_define_code:nmn {fontspec} {LetterSpace}
393 {
394   \@@_update_featstr:n {letterspace=#1}
395 }

```

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.

```

396 \@@_keys_define_code:nmn {fontspec} {HyphenChar}
397 {
398   \str_if_eq:nnTF {#1} {None}
399   {
400     \tl_put_right:Nn \l_@@_postadjust_tl
401       { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
402   }
403   {
404     \@@_warning:nx {only-xetex-feature} {HyphenChar}
405
406     \tl_if_single:nTF {#1}
407     { \tl_set:Nn \l_@@_hyphenchar_tl {`#1} }
408     { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }
409
410     \exp_args:No \@@_primitive_font_glyph_if_exist:NnTF \l_@@_fontface_cs_tl {\l_@@_hyphen
411       {
412         \tl_put_right:Nn \l_@@_postadjust_tl
413           { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
414       }
415       { \@@_error:nxx {no-glyph}{\l_fontspec_fontname_tl}{#1} }
416
417     }
418   }
419   \@@_aff_error:n {HyphenChar}

```

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

```

420 \@@_keys_define_code:nmn {fontspec} {Color}
421 {
422   \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
423   {
424     \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
425   }
426   {
427     \int_compare:nTF { \tl_count:n {#1} == 6 }
428     { \tl_set:Nn \l_@@_hexcol_tl {#1} }
429     {
430       \int_compare:nTF { \tl_count:n {#1} == 8 }
431       { \fontspec_parse_colour:viii #1 }
432       {
433         \bool_if:NF \l_@@_firsttime_bool

```

```

434             { \@@_warning:nx {bad-colour} {#1} }
435         }
436     }
437 }
438 }
439 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
440 {
441     \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
442     \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
443     {
444         \bool_if:NF \l_@@_firsttime_bool
445         { \@@_warning:nx {opa-twice-col} {#7#8} }
446     }
447     \tl_set:Nn \l_@@_opacity_tl {#7#8}
448 }
449 \aliasfontfeature{Color}{Colour}
450 \@@_keys_define_code:nnn {fontspec} {Opacity}
451 {
452     \int_set:Nn \l_@@_tmp_int {255}
453     \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
454     \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
455     {
456         \bool_if:NF \l_@@_firsttime_bool
457         { \@@_warning:nx {opa-twice} {#1} }
458     }
459     \tl_set:Nx \l_@@_opacity_tl
460     {
461         \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
462         \int_to_hex:n { \l_@@_tmp_int }
463     }
464 }

```

Mapping

```

465 < *XE>
466 \@@_keys_define_code:nnn {fontspec-aat} {Mapping}
467 {
468     \tl_set:Nn \l_@@_mapping_tl { #1 }
469 }
470 \@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
471 {
472     \tl_set:Nn \l_@@_mapping_tl { #1 }
473 }
474 < /XE>
475 < *LU>
476 \@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
477 {
478     \str_if_eq:nnTF {#1} {tex-text}
479     {
480         \@@_warning:n {no-mapping-ligtext}
481         \msg_redirect_name:nnn {fontspec} {no-mapping-ligtext} {none}

```

```

482     \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
483   }
484   { \@@_warning:n {no-mapping} }
485 }
486 </LU>

```

1.4.1 Continuous font axes

```

487 \@@_keys_define_code:nnn {fontspec} {Weight}
488 {
489   \@@_update_featstr:n{weight=#1}
490 }
491 \@@_keys_define_code:nnn {fontspec} {Width}
492 {
493   \@@_update_featstr:n{width=#1}
494 }
495 \@@_keys_define_code:nnn {fontspec} {OpticalSize}
496 <*XE>
497 {
498   \bool_if:NTF \l_@@_ot_bool
499   {
500     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
501   }
502   {
503     \bool_if:NT \l_@@_mm_bool
504     {
505       \@@_update_featstr:n { optical size = #1 }
506     }
507   }
508   \bool_if:N { !\l_@@_ot_bool && !\l_@@_mm_bool }
509   {
510     \bool_if:NT \l_@@_firsttime_bool
511     { \@@_warning:nx {no-opticals} {\l_fonts_spec_fontname_tl} }
512   }
513 }
514 </XE>
515 <*LU>
516 {
517   \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
518 }
519 </LU>

```

1.4.2 Font transformations

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

```

520 \keys_define:nn {fontspec}
521 {
522   FakeSlant .code:n =
523   {
524     \@@_update_featstr:n {slant=#1}
525   },
526   FakeSlant .default:n = {0.2}

```

```

527 }
528 \keys_define:nn {fontspec}
529 {
530   FakeStretch .code:n =
531   {
532     \@@_update_featstr:n {extend=#1}
533   },
534   FakeStretch .default:n = {1.2}
535 }
536 <*XE>
537 \keys_define:nn {fontspec}
538 {
539   FakeBold .code:n =
540   {
541     \@@_update_featstr:n {embolden=#1}
542   },
543   FakeBold .default:n = {1.5}
544 }
545 </XE>
546 <*LU>
547 \keys_define:nn {fontspec}
548 {
549   FakeBold .code:n = { \@@_warning:n {fakebold-only-xetex} }
550 }
551 </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.)

```

552 \keys_define:nn {fontspec}
553 {
554   AutoFakeSlant .code:n =
555   {
556     \bool_if:NT \l_@@_firsttime_bool
557     {
558       \tl_set:Nn \l_@@_fake_slant_tl {#1}
559       \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
560       \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
561       \bool_set_false:N \l_@@_noit_bool
562
563       \tl_if_empty:NF \l_@@_fake_embolden_tl
564       {
565         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
566         {FakeBold=\l_@@_fake_embolden_tl}
567         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
568         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
569       }

```

```

570     }
571   },
572   AutoFakeSlant .default:n = {0.2}
573 }

Same but reversed:
574 \keys_define:nn {fontspec}
575 {
576   AutoFakeBold .code:n =
577   {
578     \bool_if:NT \l_@@_firsttime_bool
579     {
580       \tl_set:Nn \l_@@_fake_embolden_tl {#1}
581       \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
582       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
583       \bool_set_false:N \l_@@_nobf_bool
584
585       \tl_if_empty:NF \l_@@_fake_slant_tl
586       {
587         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
588         {FakeSlant=\l_@@_fake_slant_tl}
589         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
590         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
591       }
592     }
593   },
594   AutoFakeBold .default:n = {1.5}
595 }

```

1.4.3 Raw feature string

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

```

596 \@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}
597 {
598   \@@_update_featstr:n {#1}
599 }
600 \@@_keys_define_code:nnn {fontspec-aat} {RawFeature}
601 {
602   \@@_update_featstr:n {#1}
603 }

```

File XIV

fontspec-code-feat-opentype.dtx

1 OpenType feature definitions

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

2 Regular key=val / tag definitions

2.1 Ligatures

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

Emulate CM extra ligatures.

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

2.2 Letters

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

```

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

```

2.3 Numbers

```

42 \@@_define_opentype_feature_group:n {Numbers}
43 \@@_define_opentype_feature:nnnnn {Numbers} {ResetAll} {} {}
44 {
45   +tnum,-tnum,
46   +pnum,-pnum,
47   +onum,-onum,
48   +lnum,-lnum,
49   +zero,-zero,
50   +anum,-anum,
51 }
52 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Monospaced} {tnum} {tnum} {+pnum,-pnum}
53 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Proportional} {pnum} {pnum} {+tnum,-tnum}
54 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Lowercase} {onum} {onum} {+lnum,-lnum}
55 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Uppercase} {lnum} {lnum} {+onum,-onum}
56 \@@_define_opentype_onoffreset:nnnnn {Numbers} {SlashedZero} {zero} {zero} {}
57 \aliasfontfeatureoption {Numbers} {Monospaced} {Tabular}
58 \aliasfontfeatureoption {Numbers} {Lowercase} {OldStyle}
59 \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.

```

60 (LU) \@@_define_opentype_onoffreset:nnnnn {Numbers} {Arabic} {anum} {anum} {}

```

2.4 Vertical position

```

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

```


2.5 Contextuals

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

2.6 Diacritics

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

2.7 Kerning

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

2.8 Fractions

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

```

119     +afrc,-afrc,
120 }

121 \@@_define_opentype_feature:nnnnn {Fractions} {0n} {frac} {+frac} {}
122 \@@_define_opentype_feature:nnnnn {Fractions} {Off} {frac} {-frac} {}
123 \@@_define_opentype_feature:nnnnn {Fractions} {Reset} {} {} {+frac,-frac}
124 \@@_define_opentype_onoffreset:nnnnn {Fractions} {Alternate} {afrc} {afrc} {-frac}

125 \@@_define_opentype_feature_group:n {LocalForms}
126 \@@_define_opentype_feature:nnnnn {LocalForms} {0n} {locl} {+locl} {}
127 \@@_define_opentype_feature:nnnnn {LocalForms} {Off} {locl} {-locl} {}
128 \@@_define_opentype_feature:nnnnn {LocalForms} {Reset} {} {} {+locl,-locl}

```

2.9 Style

```

129 \@@_define_opentype_feature_group:n {Style}
130 \@@_define_opentype_feature:nnnnn {Style} {ResetAll} {} {}
131 {
132     +salt,-salt,
133     +ital,-ital,
134     +ruby,-ruby,
135     +swsh,-swsh,
136     +hist,-hist,
137     +titl,-titl,
138     +hkna,-hkna,
139     +vkna,-vkna,
140     +ssty=0,-ssty=0,
141     +ssty=1,-ssty=1,
142 }

143 \@@_define_opentype_onoffreset:nnnnn {Style} {Alternate} {salt} {salt} {}
144 \@@_define_opentype_onoffreset:nnnnn {Style} {Italic} {ital} {ital} {}
145 \@@_define_opentype_onoffreset:nnnnn {Style} {Ruby} {ruby} {ruby} {}
146 \@@_define_opentype_onoffreset:nnnnn {Style} {Swash} {swsh} {swsh} {}
147 \@@_define_opentype_onoffreset:nnnnn {Style} {Cursive} {swsh} {curs} {}
148 \@@_define_opentype_onoffreset:nnnnn {Style} {Historic} {hist} {hist} {}
149 \@@_define_opentype_onoffreset:nnnnn {Style} {TitlingCaps} {titl} {titl} {}
150 \@@_define_opentype_onoffreset:nnnnn {Style} {HorizontalKana} {hkna} {hkna} {+vkna,+pkna}
151 \@@_define_opentype_onoffreset:nnnnn {Style} {VerticalKana} {vkna} {vkna} {+hkna,+pkna}
152 \@@_define_opentype_onoffreset:nnnnn {Style} {ProportionalKana} {pkna} {pkna} {+vkna,+hkna}
153 \@@_define_opentype_feature:nnnnn {Style} {MathScript} {ssty} {+ssty=0} {+ssty=1}
154 \@@_define_opentype_feature:nnnnn {Style} {MathScriptScript} {ssty} {+ssty=1} {+ssty=0}

```

2.10 CJK shape

```

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

```

```

165     +nlck,-nlck,
166 }

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

```

2.11 Character width

```

175 \@@_define_opentype_feature_group:n {CharacterWidth}
176 \@@_define_opentype_feature:nnnnn {CharacterWidth} {ResetAll} {} {}
177 {
178     +pwid,-pwid,
179     +fwid,-fwid,
180     +hwid,-hwid,
181     +twid,-twid,
182     +qwid,-qwid,
183     +palt,-palt,
184     +halt,-halt,
185 }

186 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Proportional} {pwid} {pwid} {+}
187 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Full} {fwid} {fwid} {+}
188 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Half} {hwid} {hwid} {+}
189 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Third} {twid} {twid} {+}
190 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Quarter} {qwid} {qwid} {+}
191 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateProportional} {palt} {palt} {+}
192 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateHalf} {halt} {halt} {+}

```

2.12 Vertical

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

```

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

```

3 OpenType features that need numbering

3.1 Alternate

```

202 \@@_define_opentype_feature_group:n {Alternate}

```

```

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

```

3.2 Variant / StylisticSet

```

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

```

3.3 CharacterVariant

```

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

```

```

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

```

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

3.4 Annotation

```

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

```

3.5 Ornament

```

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

```

4 Script and Language

4.1 Script

```

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

```

```

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

```

4.2 Language

```

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

```

```

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

```

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

```

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

```

5 Backwards compatibility

```

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

```

File XV

fontspec-code-scripts.dtx

1 Font script definitions

```
1 \newfontscript{Adlam}{adlm}
2 \newfontscript{Ahom}{ahom}
3 \newfontscript{Anatolian~Hieroglyphs}{hluw}
4 \newfontscript{Arabic}{arab}
5 \newfontscript{Armenian}{armn}
6 \newfontscript{Avestan}{avst}
7 \newfontscript{Balinese}{bali}
8 \newfontscript{Bamum}{bamu}
9 \newfontscript{Bassa~Vah}{bass}
10 \newfontscript{Batak}{batk}
11 \newfontscript{Bengali}{bng2,beng}
12 \newfontscript{Bhaiksuki}{bhks}
13 \newfontscript{Bopomofo}{bopo}
14 \newfontscript{Brahmi}{brah}
15 \newfontscript{Braille}{brai}
16 \newfontscript{Buginese}{bugi}
17 \newfontscript{Buhid}{buhd}
18 \newfontscript{Byzantine~Music}{byzm}
19 \newfontscript{Canadian~Syllabics}{cans}
20 \newfontscript{Carian}{cari}
21 \newfontscript{Caucasian~Albanian}{aghb}
22 \newfontscript{Chakma}{cakm}
23 \newfontscript{Cham}{cham}
24 \newfontscript{Cherokee}{cher}
25 \newfontscript{CJK~Ideographic}{hani}
26 \newfontscript{Coptic}{copt}
27 \newfontscript{Cypriot~Syllabary}{cpri}
28 \newfontscript{Cyrillic}{cyrl}
29 \newfontscript{Default}{DFLT}
30 \newfontscript{CustomDefault}{latn,DFLT}
31 \newfontscript{Deseret}{dsrt}
32 \newfontscript{Devanagari}{dev2,deva}
33 \newfontscript{Duployan}{dupl}
34 \newfontscript{Egyptian~Hieroglyphs}{egyp}
35 \newfontscript{Elbasan}{elba}
36 \newfontscript{Ethiopic}{ethi}
37 \newfontscript{Georgian}{geor}
38 \newfontscript{Glagolitic}{glag}
39 \newfontscript{Gothic}{goth}
40 \newfontscript{Grantha}{gran}
41 \newfontscript{Greek}{grek}
42 \newfontscript{Gujarati}{gjr2,gujr}
43 \newfontscript{Gurmukhi}{gur2,guru}
44 \newfontscript{Hangul~Jamo}{jamo}
```



```

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

```

```

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

```

For convenience or backwards compatibility:

```

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

```

File XVI

fontspec-code-lang.dtx

1 Font language definitions

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

```

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

```

96 \newfontlanguage{French}{FRA}
 97 \newfontlanguage{Frisian}{FRI}
 98 \newfontlanguage{Friulian}{FRL}
 99 \newfontlanguage{Futa}{FTA}
 100 \newfontlanguage{Fulani}{FUL}
 101 \newfontlanguage{Ga}{GAD}
 102 \newfontlanguage{Gaelic}{GAE}
 103 \newfontlanguage{Gagauz}{GAG}
 104 \newfontlanguage{Galician}{GAL}
 105 \newfontlanguage{Garshuni}{GAR}
 106 \newfontlanguage{Garhwali}{GAW}
 107 \newfontlanguage{Ge'ez}{GEZ}
 108 \newfontlanguage{Gilyak}{GIL}
 109 \newfontlanguage{Gumuz}{GMZ}
 110 \newfontlanguage{Gondi}{GON}
 111 \newfontlanguage{Greenlandic}{GRN}
 112 \newfontlanguage{Garó}{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}{K RK}
¹⁸² \newfontlanguage{Karelian}{KRL}
¹⁸³ \newfontlanguage{Karaim}{KRM}
¹⁸⁴ \newfontlanguage{Karen}{KRN}
¹⁸⁵ \newfontlanguage{Koorete}{KRT}
¹⁸⁶ \newfontlanguage{Kashmiri}{KSH}
¹⁸⁷ \newfontlanguage{Khasi}{KSI}
¹⁸⁸ \newfontlanguage{Kildin~Sami}{KSM}
¹⁸⁹ \newfontlanguage{Kui}{KUI}
¹⁹⁰ \newfontlanguage{Kulvi}{KUL}
¹⁹¹ \newfontlanguage{Kumyk}{KUM}
¹⁹² \newfontlanguage{Kurdish}{KUR}
¹⁹³ \newfontlanguage{Kurukh}{KUU}
¹⁹⁴ \newfontlanguage{Kuy}{KUY}
¹⁹⁵ \newfontlanguage{Koryak}{KYK}
¹⁹⁶ \newfontlanguage{Ladin}{LAD}
¹⁹⁷ \newfontlanguage{Lahuli}{LAH}

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

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

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

```

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

```

File XVII

fontspec-code-feat-aat.dtx

1 AAT feature definitions

These are only defined for X_YTeX.

1.1 Ligatures

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

Emulate CM extra ligatures.

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

1.2 Letters

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

1.3 Numbers

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

Both naming conventions are offered to select the number case.

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

1.4 Contextuals

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

1.5 Diacritics

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

1.6 Vertical position

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

1.7 Fractions

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

1.8 Alternate

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

```

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

```

1.9 Variant / StylisticSet

```

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

```

1.10 Style

```

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

```

1.11 CJK shape

```

107 \@@_define_aat_feature_group:n {CJKShape}
108 \@@_define_aat_feature:nnnn {CJKShape} {Traditional} {20} {0}
109 \@@_define_aat_feature:nnnn {CJKShape} {Simplified} {20} {1}
110 \@@_define_aat_feature:nnnn {CJKShape} {JIS1978} {20} {2}
111 \@@_define_aat_feature:nnnn {CJKShape} {JIS1983} {20} {3}
112 \@@_define_aat_feature:nnnn {CJKShape} {JIS1990} {20} {4}

```

```

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

```

1.12 Character width

```

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

```

1.13 Annotation

```

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

```

File XVIII

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

```

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

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

\UndeclareAccent 64 \DeclareDocumentCommand \UndeclareSymbol {m}

```

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

```



```

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

```

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

\UndeclareComposite

```

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

```

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

File XIX

fontspec-code-math.dtx

1 Selecting maths fonts

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

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

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

```

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

```

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

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

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

```

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

```

```

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

```

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

File XX

fontspec-code-closing.dtx

1 Closing code

1.1 Finishing up

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

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

File XXI

fontspec-code-xfss.dtx

1 Changes to the NFSS

1 `<*fontspec>`

1.1 Italic small caps and so on

`\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_{TEX}'s 'shape' font axis needs to be overloaded to support italic small caps and slanted small caps. These are the combinations to support:

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

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

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

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

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



```

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

```

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

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

```

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

```

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

1.2 Emphasis

`\emfontdeclare`

```

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

```

```

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

```

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

`\em`

```

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

```

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

`\emph`

```

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

```

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

1.3 Strong emphasis

`\strongfontdeclare`

```

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

```

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

`\strongenv`

```

130 \DeclareRobustCommand \strongenv
131 {
132   \@nomath\strongenv
133
134   <debug> \typeout{Strong~ level:~\int_use:N \l_@@_strong_int}
135   \prop_get:NxNT \g_@@_strong_prop { \f@series } \l_@@_strong_tmp_tl
136   {
137     \int_set:Nn \l_@@_strong_int { \l_@@_strong_tmp_tl }
138   <debug> \typeout{Series~ (\f@series)~ detected;~ new~ level:~\int_use:N \l_@@_strong_int}
139   }
140
141   \int_incr:N \l_@@_strong_int
142
143   \prop_get:NxNTF \g_@@_strong_prop { switch-\int_use:N \l_@@_strong_int } \l_@@_strong_switch_tl
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-code-patches.dtx

1 Patching code

1 `<*fontspec>`

1.1 Verbatims

Many thanks to Apostolos Syropoulos for discovering this problem and writing the redefinition of L^AT_EX's verbatim environment and `\verb*` command.

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

```
2 \cs_new_protected:Nn \fontspec_visible_space:
3 {
4   \@@_primitive_font_glyph_if_exist:NnTF \font {"2423}
5     { \char"2423\scan_stop: }
6     { \fontspec_visible_space_fallback: }
7 }
```

(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.

```
8 \cs_new_protected:Nn \fontspec_visible_space_fallback:
9 {
10   {
11     \usefont{\g_fontspec_encoding_tl}{lmtt}{\f@series}{\f@shape}
12     \textvisiblespace
13   }
14 }
```

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

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

```
15 \group_begin:
16 \char_set_catcode_active:n{"20}%
17 \cs_gset:Npn\fontspec_print_visible_spaces:{%
18 \char_set_catcode_active:n{"20}%
19 \cs_set_eq:NN^~20\fontspec_visible_space:%
20 }%
21 \group_end:
```

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

In 2019 there will be a new kernel-supported command for generalising 'verbatim visible spaces'. For now we check if the new command is defined and disable all patching if so. In 2020, I suppose, I'll remove all the patching code here.

```
22 \AtBeginDocument
23 {
24   \cs_if_exist:NF \verbvisiblespace
25   {
```

```

26 \@@_patch_verb:
27 \@@_patch_verbatim:
28 \@@_patch_moreverb:
29 \@@_patch_fancyvrb:
30 \@@_patch_listings:
31 }
32 }

```

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

```

\verb* 33 \cs_new_protected:Npn \@@_patch_verb:
34 {
35   \def\verb
36   {
37     \relax\ifmmode\hbox\else\leavevmode\null\fi
38     \bgroup
39     \verb@eol@error \let\do\@makeoother \dospecials
40     \verbatim@font\@noligs
41     \@ifstar\@@sverb\@verb
42   }
43   \def\@@sverb{\fontspec_print_visible_spaces:\@sverb}
44 }

```

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

`verbatim*` With the `verbatim` package.

```

45 \cs_new_protected:Npn \@@_patch_verbatim:
46 {
47   \@ifpackageloaded{verbatim}
48   {
49     \cs_set:cpn {verbatim*}
50     {
51       \group_begin: \@verbatim \fontspec_print_visible_spaces: \verbatim@start
52     }
53   }

```

This is for vanilla \LaTeX .

```

54   {
55     \cs_set:cpn {verbatim*}
56     {
57       \@verbatim \fontspec_print_visible_spaces: \@sxverbatim
58     }
59   }
60 }

```

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

```

61 \cs_new_protected:Npn \@@_patch_moreverb:
62 {
63   \@ifpackageloaded{moreverb}
64   {
65     \cs_set:cpn {listingcont*}
66     {
67       \cs_set:Npn \verbatim@processline

```

```

68         {
69             \thelisting@line \global\advance\listing@line1\relax
70             \the\verbatim@line\par
71         }
72         \@verbatim \fontspec_print_visible_spaces: \verbatim@start
73     }
74 }{}
75 }

```

listings and fancvrb make things nice and easy:

```

76 \cs_new_protected:Npn \@@_patch_fancyvrb:
77 {
78     \@ifpackageloaded{fancyvrb}
79     {
80         \cs_set_eq:NN \FancyVerbSpace \fontspec_visible_space:
81     }{}
82 }
83 \cs_new_protected:Npn \@@_patch_listings:
84 {
85     \@ifpackageloaded{listings}
86     {
87         \cs_set_eq:NN \lst@visiblespace \fontspec_visible_space:
88     }{}
89 }
90 </fontspec>

```

Index

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

Symbols	
\#	250, 251, 282
\,	1, 2, 3, 4, 5, 17
\-	653
@@ commands:	
\@@_DeclareFontShape:nnnnnn	498, 505, <u>515</u> , 533
\g_@@_OT_features_prop	9, 11, 71
\@@_add_nfssfont:nnnn	285, 305, 306, 307, 308, 309, 310, <u>325</u>
\@@_aff_error:n	11, 346, 387, 419
\l_@@_alias_bool	22, 204, 211, 217, 222, 229, 249
\l_@@_all_features_clist	21, 53, 99, 109, 123, 186, 271
\g_@@_all_keyval_modules_clist	1, 47, 206, 224
\g_@@_all_opentype_feature_names_prop	72, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343
\l_@@_arg_clist	59, 272, 273, 274, 277, 280
\l_@@_atsui_bool	7, 10, 204, 344, 353, 610
\l_@@_basename_tl	10, 42, 88, 322
\l_@@_bf_series_seq	45, 148, 160, 163
\g_@@_bfmathrm_tl	69, 70, 82, 87, 88, 89, 118
\@@_calc_scale:n	292, 293, <u>303</u>
\g_@@_cfg_bool	1, 7, 8, 17
\l_@@_check_bool	5, 65, 66, 195, 200, 206, 217
\@@_check_lang:Nn	121
\@@_check_lang:NnTF	97, <u>121</u> , 336
\@@_check_lang:Nnn	125
\@@_check_lang:NnnTF	110, <u>121</u>
\@@_check_ot_feat:Nn	169
\@@_check_ot_feat:NnTF	50, 60, <u>169</u> , 602
\@@_check_ot_feat:Nnnn	174
\@@_check_ot_feat:NnnnTF	67, <u>169</u>
\@@_check_script:Nn	75
\@@_check_script:NnTF	75, 80, 290
\@@_combo_sc_shape:n	506, 509, 554, 562
\@@_construct_font_call:nn	136, 138, 142, 145, 151, 176, 274, 391, 392, 413, 492
\@@_construct_font_call:nnnnnn	151, 160
\l_@@_curr_bfname_tl	91, 158, 168, 171, 173, 205
\l_@@_curr_fontname_tl	90, 316, 317
\g_@@_curr_series_tl	89, 147, 162, 166, 171, 173, 205, 649
\@@_declare_shape:nnnn	402, <u>421</u>
\@@_declare_shape_loginfo:nn	433, <u>537</u>
\@@_declare_shape_slanted:nn	432, <u>525</u>
\@@_declare_shapes_normal:nn	430, <u>496</u>
\@@_declare_shapes_smcaps:nn	431, <u>501</u>
\g_@@_default_fontopts_clist	46, 111, 119
\@@_define_aat_feature:nnnn	2, 3, 4, <u>5</u> , 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 60, 61, 62, 63, 65, 66, 67, 101, 102, 103, 104, 105, 106, 108, 109, 110, 111, 112, 113, 114, 116, 117, 118, 119, 120, 121, 122, 123, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 183
\@@_define_aat_feature_group:n	1, 1, 29, 35, 44, 55, 59, 64, 68, 80, 91, 100, 107, 115, 124, 178

\@@_define_opentype_feature:nnnnn	\l_@@_family_fontopts_clist
. 5, 7, 30, 43, 52, 105, 106, 112
43, 44, 45, 49, 50, 62, 78, 94, 106, 112,	\g_@@_family_int_prop . . . 76, 248, 254
113, 114, 116, 121, 122, 123, 126,	\l_@@_family_label_tl
127, 128, 130, 153, 154, 156, 176, 193 105, 107, 124, 150, 164
\@@_define_opentype_feature_-	\@@_feat_off:n 39, 44
group:n 1, 6, 29, 42,	\@@_feat_prop_add:nn . 1, 2, 3, 4, 5, 5, 17
61, 77, 93, 105, 115, 125, 129, 155,	\@@_feat_reset:n 40, 45, 50
175, 188, 193, 202, 221, 235, 257, 267	\@@_find_autofonts: 262, 282
\@@_define_opentype_onoffreset:nnnnn	\l_@@_firsttime_bool
. 13, 14, 15, 16, 17, 18, 1, 29, 179, 225, 271, 343,
35, 36, 37, 38, 39, 39, 40, 41, 52, 53,	433, 444, 456, 510, 556, 578, 621, 642
54, 55, 56, 60, 71, 72, 73, 74, 75, 76,	\@@_font_is_file: 30, 168, 175
87, 88, 89, 90, 91, 92, 101, 102, 103,	\@@_font_is_name: 168, 643
104, 111, 124, 143, 144, 145, 146,	\l_@@_font_path_tl . . . 29, 94, 174, 644
147, 148, 149, 150, 151, 152, 167,	\@@_font_suppress_not_found_-
168, 169, 170, 171, 172, 173, 174,	error: 5, 9, 38, 267
186, 187, 188, 189, 190, 191, 192,	\l_@@_fontcfg_bool . . . 12, 13, 18, 22, 81
194, 195, 196, 197, 198, 199, 200, 201	\l_@@_fontface_cs_tl 17,
\@@_define_opentype_onreset:nnnnn	60, 144, 148, 149, 154, 159, 160, 163,
. 28, 47	167, 290, 312, 336, 410, 412, 602, 613
\g_@@_defined_shapes_tl	\l_@@_fontfeat_bf_clist
. 86, 189, 539, 648 62, 202, 306, 581
\l_@@_defining_encoding_bool . 11,	\l_@@_fontfeat_bfit_clist
17, 23, 24, 29, 35, 59, 61, 66, 72, 78, 84 64, 213, 309, 565, 567, 587, 589
\l_@@_disable_defaults_bool 21, 97, 153	\l_@@_fontfeat_bfsl_clist 66, 221, 310
\l_@@_em_int	\l_@@_fontfeat_clist 57, 130, 193, 272
. 36, 93, 96, 97, 100, 102, 105, 156, 159	\l_@@_fontfeat_curr_clist
\g_@@_em_normalise_slant_bool 58, 452, 461, 474
. 26, 62, 68, 88	\l_@@_fontfeat_it_clist
\g_@@_em_prop . . . 60, 73, 77, 78, 94, 102 63, 209, 307, 559
\l_@@_em_switch_tl 102, 103, 110	\l_@@_fontfeat_sc_clist . 67, 227, 452
\l_@@_em_tmp_tl 94, 96, 115	\l_@@_fontfeat_sl_clist . 65, 217, 308
\l_@@_emdef_int 37, 61, 77, 78, 79	\l_@@_fontfeat_up_clist
\l_@@_emshape_query_tl 61, 198, 233, 305
. 86, 90, 94, 97, 109	\g_@@_fontid_family_prop 75, 228, 256
\@@_error:n 1, 10, 443	\l_@@_fontid_tl
\@@_error:nn 2, 3, 12, 14, 18, 24, 21, 23, 42, 95, 224, 228, 236
30, 36, 67, 73, 79, 85, 140, 414, 684, 703	\l_@@_fontname_bf_tl
\@@_error:nnn 415 75, 128, 168, 287, 293, 306, 582
\g_@@_euenc_bool 9, 10, 19, 23, 36, 39, 58	\l_@@_fontname_bfit_tl 76,
\l_@@_ext_filename_tl 85, 86, 89, 90, 92	130, 179, 286, 287, 288, 309, 568, 590
\l_@@_extension_tl	\l_@@_fontname_bfsl_tl
. 39, 45, 53, 72, 93, 116, 162 132, 183, 301, 310
\l_@@_extensions_clist 48, 50, 61, 254	\l_@@_fontname_it_tl
\l_@@_external_bool . . . 23, 28, 40, 375 74, 129, 138, 286, 298, 307, 560
\@@_extract_all_features: 94	\l_@@_fontname_sc_tl 133, 193, 456, 468
\@@_extract_all_features:n . . . 20, 94	\l_@@_fontname_sl_tl 131, 143, 301, 308
\l_@@_fake_italic_tl	\l_@@_fontname_tl . . . 96, 152, 154, 158
. 126, 563, 566, 580	\l_@@_fontname_up_tl 9, 42,
\l_@@_fake_slant_tl . 125, 558, 585, 588	45, 126, 127, 128, 136, 138, 140, 142, 145

\@@_fontname_wrap:n	\l_@@_lang_tl
..... 47, 153, 154, 170, 174	.. 48, 137, 171, 280, 338, 355, 580, 595
\l_@@_fontopts_clist	\l_@@_language_int
.. 51, 102, 103, 113, 402, 409, 410, 411	30, 45, 185, 186, 192, 198, 278, 339, 356
\g_@@_fontopts_prop	\l_@@_leftover_clist 54, 400, 402
68, 87, 102, 105, 132, 135, 139, 140, 409	\@@_load_external_fontoptions:Nn
\@@_get_features:Nn 18, 79, 408
59, 189	\@@_load_font:
\@@_get_features:n .. 28, 189, 273, 481 26, 132
\l_@@_graphite_bool . 11, 204, 347, 365	\@@_load_fontname:Nn 401, 405, 447, 468
\l_@@_harfbuzz_bool	\@@_lua_function:ne
10, 74, 97	78
\c_@@_hexcol_tl	\@@_lua_function:nee
149, 217, 664	78
\l_@@_hexcol_tl	\@@_lua_function:neee
..... 142, 216, 218, 424, 428, 441, 664	78, 163
\l_@@_hyphenchar_tl	\@@_lua_function:neeee
..... 141, 407, 408, 410, 413	78, 216
\@@_if_autofont:nn	\@@_main_DeclareFontExtensions:n
388 122, 252
\@@_if_autofont:nnTF	\@@_main_IfFontFeatureActiveTF:nnn
381 126, 257
\@@_if_detect_external:n	\@@_main_addfontfeatures:n 82, 86, 144
58	\@@_main_aliasfontfeature:nn 106, 201
\@@_if_detect_external:nTF 12, 58, 175	\@@_main_aliasfontfeatureoption:nnn
\@@_if_font_feature:n 110, 220
263	\@@_main_fontspec:nn
\@@_if_font_feature:nTF	1, 3
261	\@@_main_liningnums:n
\@@_if_merge_shape:n	137, 292
25	\@@_main_newAATfeature:nnnn .. 94, 175
\@@_if_merge_shape:nTF	\@@_main_newfontface:NnnN
21, 191 59, 63, 67, 71, 112
\@@_info:n	\@@_main_newfontfamily:NnnN
7, 133, 301, 320 43, 47, 51, 55, 99, 114
\@@_info:nn	\@@_main_newfontfeature:nn ... 90, 168
8, 383	\@@_main_newopentypefeature:nnn
\@@_info:nnn 98, 102, 185
9, 265	\@@_main_oldstylenums:n 132, 285
\@@_init:	\@@_main_setboldmathrm:nn ... 27, 67
6, 174, 268, 638	\@@_main_setmainfont:nn
\@@_init_fontface:	7, 8, 39
192, 659	\@@_main_setmathrm:nn
\@@_init_ttc:n	23, 61
17, 70	\@@_main_setmathsf:nn
\@@_int_mult_truncate:Nn 74, 453	31, 73
\@@_iv_str_to_num:Nn	\@@_main_setmathtt:nn
..... 86, 135, 136, 184, 185, 187, 687	35, 79
\@@_iv_str_to_num:w 697, 698, 700	\@@_main_setmonofont:nn
\@@_keys_define_code:nnn 7, 13, 16,	13, 25
20, 24, 36, 37, 46, 104, 109, 114, 121,	\@@_make_AAT_feature:nn . 9, 12, 76, 87
126, 130, 141, 145, 150, 177, 181,	\@@_make_AAT_feature_string:Nnn . 26
185, 196, 200, 207, 211, 215, 219,	\@@_make_AAT_feature_string:NnnTF
223, 230, 235, 241, 245, 249, 253, 12, 17, 26, 612
257, 261, 265, 269, 288, 297, 341,	\@@_make_OT_feature:nnn
353, 367, 388, 392, 396, 420, 450,	33, 52, 71, 210, 217, 230, 241, 264, 274
466, 470, 476, 487, 491, 495, 596, 600	\@@_make_font_shapes:Nnnnn .. 317, 397
\l_@@_keys_leftover_clist	\@@_make_ot_smallcaps:TF
..... 55, 124, 127, 128, 129,	599
194, 195, 199, 200, 203, 205, 209, 210	\@@_make_smallcaps:TF .. 458, 599, 605
\@@_keys_set_known:nnN	\l_@@_mapping_tl
.. 67, 122, 127, 129, 193, 195, 331, 400 22, 24, 26, 145, 213, 214, 468, 472
\@@_lang_dflt_correct:N 186, 714	
\l_@@_lang_name_tl	
..... 112, 131, 138, 139, 183, 185	

\g_@@_math_bool	\@@_ot_compat:nn ...
... 5, 6, 18, 98, 100, 101, 102, 103,	359, 363, 364,
104, 105, 106, 107, 110, 112, 113,	365, 366, 367, 368, 369, 370, 371,
116, 118, 119, 120, 121, 122, 123,	372, 373, 374, 375, 376, 377, 378, 379
124, 125, 126, 127, 128, 129, 130, 131	\@@_ot_validate_tag:n
\g_@@_math_euler_bool 102, 158, 159, 210, 211, 212, 669
9, 14, 34	\@@_ot_validate_tag:w
\g_@@_math_lucida_bool	672, 675
... 13, 14, 15, 15, 40	\@@_ot_validate_tag_aux:w 678, 679, 681
\g_@@_mathrm_tl	\@@_patch_fancyvrb:
63,	29, 76
64, 74, 75, 77, 78, 81, 84, 96, 117, 121	\@@_patch_listings:
\g_@@_mathsf_tl	30, 83
... 75, 76, 79, 91, 97, 119, 122	\@@_patch_moreverb:
\g_@@_mathtt_tl	28, 61
... 80, 81, 82, 92, 98, 120, 123	\@@_patch_verb:
\l_@@_mm_bool ... 9, 346, 357, 503, 508	26, 33
\l_@@_mode_tl 76, 84, 86, 90, 99, 587, 652	\@@_patch_verbatim:
\@@_msg_new:nnn	27, 45
13,	\g_@@_pkg_euler_loaded_bool 2, 3, 8, 16
18, 23, 44, 84, 90, 94, 104, 108, 112,	\c_@@_postadjust_tl
117, 122, 127, 133, 137, 143, 147,	151, 665
151, 156, 160, 165, 169, 174, 179,	\l_@@_postadjust_tl
183, 191, 195, 199, 203, 207, 212, 217	148, 390,
\@@_msg_new:nnnn 15, 28, 37, 48, 58, 66, 74	400, 412, 499, 506, 534, 565, 568, 665
\l_@@_never_check_bool	\l_@@_pre_feat_sclist 153, 275, 493, 572
... 28, 78, 128, 176, 270	\@@_preparse_features:
\g_@@_nfss_enc_tl 4, 15, 21, 33, 39, 51,	25, 118
57, 99, 107, 263, 263, 498, 505, 533, 650	\l_@@_prev_unicode_name_tl ...
\l_@@_nfss_fam_tl .. 103, 226, 241, 267	57, 62
\g_@@_nfss_family_tl	\l_@@_primitive_font
... 41, 100, 161, 185, 230, 241, 242,	39, 40
255, 256, 263, 269, 270, 271, 272,	\@@_primitive_font_current_name:
277, 278, 279, 280, 498, 505, 533, 534	... 56, 184, 186
\l_@@_nfss_prop	\@@_primitive_font_get_name:N 56, 393
69, 171, 204	\@@_primitive_font_glyph_if-
\l_@@_nfss_sc_tl	exist:Nn
... 101, 425, 473, 503, 506, 564	44
\l_@@_nfss_tl .. 102, 424, 448, 499, 552	\@@_primitive_font_glyph_if-
\l_@@_nfssfont_prop	exist:NnTF
70, 312, 335	4, 44, 410
\l_@@_nobf_bool	\@@_primitive_font_gset:Nnn
... 2, 26, 154, 157, 284, 291, 583	... 1, 26, 28, 33
\l_@@_noit_bool	\@@_primitive_font_gset:NnnTF ...
... 3, 27, 134, 137, 284, 296, 561	34
\l_@@_nosc_bool 4, 189, 192, 454, 465, 471	\@@_primitive_font_gset:NnnTFTF .
\c_@@_opacity_tl 150, 217, 442, 454, 663	21
\l_@@_opacity_tl	\@@_primitive_font_gset:Onn ..
... 143, 216, 218, 442, 447, 454, 459, 663	33, 144
\l_@@_optical_size_tl	\@@_primitive_font_gset:OnnTF ...
... 144, 165, 500, 517, 645	34
\l_@@_options_tl ... 97, 151, 154, 158	\@@_primitive_font_gset:p:NnnTF .
\l_@@_ot_bool	21
8, 28, 39,	\@@_primitive_font_if_exist:nTF
65, 78, 91, 108, 121, 136, 197, 269,	... 35, 176
345, 361, 370, 498, 508, 575, 607, 641	\@@_primitive_font_if_null:NnTF .
	... 13, 24, 29, 40
	\@@_primitive_font_if_null_p:N ..
	13
	\@@_primitive_font_set:Nnn
	... 1, 21, 23, 31, 39, 391, 392
	\@@_primitive_font_set:NnnTF .
	32, 137
	\@@_primitive_font_set:NnnTFTF ..
	21
	\@@_primitive_font_set:Onn
	31
	\@@_primitive_font_set:OnnTF .
	32, 412
	\@@_primitive_font_set_hyphenchar:Nn
	... 52, 401, 413
	\@@_primitive_font_set_p:NnnTF ..
	21
	\l_@@_proceed_bool
	27, 56, 63, 69
	\l_@@_punctspace_adjust_tl
	... 146, 151, 373, 378, 383, 667

\g_@@_rawfeatures_sclist	58, 152, 275, 276, 482, 493, 625, 634, 661	\@@_shape_merge:nn 10, 11, 12, 13, 14, 15, 16, 17, 18, 22, 28, 33, 193, 511, 512
\@@_remove_clashing_featstr:n . .	23, 67, 628	\l_@@_shaper_tl
\l_@@_renderer_tl	52, 55, 59, 60, 83, 164, 354, 362, 366, 647	81, 85, 86, 91, 100, 588, 590
\l_@@_rmfamily_family_tl 9, 10, 16, 154		\g_@@_single_feat_tl 60, 87, 88, 265, 277, 279, 281, 297, 340, 357, 623
\@@_sanitise_fontname:Nn	8, 9, 10, 44, 74, 75, 76, 84, 128	\l_@@_size_tl
\@@_save_family:nn	33, 259	104, 255, 438, 443, 444, 479, 491
\@@_save_family_needed:n	220	\l_@@_sizedfont_tl
\@@_save_family_needed:nTF . . . 31, 220		105, 259, 439, 447, 449
\@@_save_fontid_family:nn 236, 246, 258		\l_@@_sizefeat_clist
\@@_save_fontinfo:n	261, 267	48, 49, 232, 237, 330, 336
\l_@@_saved_fontname_tl . 98, 426, 439		\l_@@_sizing_leftover_clist
\l_@@_scale_tl 140, 197, 295, 299, 300, 314, 323, 484, 486, 491, 662		56, 442, 448, 474
\l_@@_script_int	29, 42, 94, 136, 138, 144, 187, 191, 198, 277, 295	\l_@@_smcp_shape_tl
\l_@@_script_name_tl . . . 107, 131, 135, 136, 141, 181, 183, 184, 293, 307		108, 193, 196, 199, 202
\l_@@_script_tl 47, 95, 123, 134, 171, 279, 294, 577, 579, 592, 594		\@@_strip_leading_sign:Nw . . 691, 694
\l_@@_scriptlang_exist_bool	25, 287, 296, 302, 333, 341, 345	\@@_strip_plus_minus:n 194, 196
\@@_select_font_family:nn	1, 43, 151, 154, 157, 165	\@@_strip_plus_minus_aux:Nq . 196, 197
\@@_set_autofont:Nnn	286, 287, 288, 293, 298, 301, 373	\l_@@_strnum_int
\@@_set_default_features:nn . . 76, 116		31, 86, 92, 135, 146, 184, 199, 295, 339
\@@_set_faces:	264, 303	\l_@@_strong_int
\@@_set_faces_aux:nnnnn . . . 312, 314		134, 137, 138, 141, 143, 146, 157, 160
\@@_set_family:NnnN . . . 147, 156, 157		\g_@@_strong_prop
\@@_set_font_default_features:nnn	77, 121	74, 116, 124, 125, 135, 143
\@@_set_font_dimen:NnN . 311, 312, 325		\l_@@_strong_switch_tl . 111, 143, 144
\@@_set_font_type:N	9, 27, 38, 64, 77, 90, 107, 120, 135, 143, 339	\l_@@_strong_tmp_tl . . . 116, 135, 137
\@@_set_fontface:NnnN . 161, 169, 170		\l_@@_strongdef_int
\@@_set_scriptlang:	27, 176	39, 117, 124, 125, 126
\@@_setboldmathrm_hook:nn . . . 71, 91		\@@_swap_plus_minus:n
\@@_setmainfont_hook:nn 22, 85		67, 72
\@@_setmathrm_hook:nn	65, 88	\@@_swap_plus_minus_aux:Nq . . 72, 73
\@@_setmathsf_hook:nn	77, 89	\l_@@_test_font
\@@_setmathtt_hook:nn	83, 90	137, 143
\@@_setmonofont_hook:nn 58, 87		\l_@@_tfm_bool
\@@_setsansfont_hook:nn 40, 86		6, 343, 350
\@@_setup_nfss:Nnnn . . . 448, 473, 477		\l_@@_this_feat_clist 60, 273, 281, 286
\@@_setup_single_size:nn . . 428, 436		\l_@@_this_font_tl 106, 238, 239, 243, 271, 280, 286, 327, 333, 336
\l_@@_sffamily_family_tl 27, 28, 34, 155		\@@_tl_new_if_free:N
		146, 152
		\l_@@_tmp_int . . . 32, 452, 453, 461, 462
		\l_@@_tmp_tl
		41, 42, 44, 45, 93, 94, 104, 105, 106, 107, 112, 123, 124, 127, 128, 132, 135, 138, 139, 139, 140, 150, 161, 162, 163, 214, 215, 216, 228, 230, 234, 235, 236, 248, 250, 251, 253, 254, 255, 331
		\l_@@_tmpa_bool
		20, 63, 66, 68
		\l_@@_tmpa_dim
		42, 311, 316
		\l_@@_tmpa_font
		391, 393
		\l_@@_tmpa_fp
		40
		\l_@@_tmpa_int 33, 139, 141, 144, 149, 152
		\l_@@_tmpa_tl
		28, 29, 53, 113
		\l_@@_tmpb_dim
		43, 312, 317

<code>\clist_put_right:Nn</code>	<code>\DeclareDocumentCommand</code>
.... 233, 559, 565, 567, 581, 587, 589	9, 15, 21, 27, 33, 39, 51, 64, 67, 70, 76, 82
<code>\clist_set:Nn</code>	<code>\DeclareErrorFont</code>
... 49, 99, 109, 198, 202, 209, 213,	42
217, 221, 227, 232, 237, 254, 272, 330	<code>\DeclareFontEncoding</code>
<code>\clist_set_eq:NN</code>	30, 41
273, 452	<code>\DeclareFontExtensions</code>
<code>\colon</code>	120
30, 31, 115	<code>\DeclareFontFamily</code>
<code>\convertcolorspec</code>	44, 263
424	<code>\DeclareFontShape</code>
cs commands: 46, 48, 50, 52, 53, 54, 54, 522
<code>\cs:w</code>	<code>\DeclareFontSubstitution</code>
127	31, 43
<code>\cs_end:</code>	<code>\DeclareMathAccent</code>
127	... 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
<code>\cs_generate_variant:Nn</code>	<code>\DeclareMathDelimiter</code> ...
..... 11, 12, 71, 73, 84, 85, 86, 87,	65, 66, 67, 68, 69
88, 89, 90, 91, 92, 93, 94, 95, 96, 97,	<code>\DeclareMathSymbol</code>
98, 99, 158, 258, 435, 524, 637, 674, 693 31, 36, 37, 38, 39, 42, 43, 44,
<code>\cs_gset:Npn</code>	45, 46, 47, 48, 49, 50, 51, 52, 53, 54,
17	55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 70
<code>\cs_if_eq:NNTF</code>	<code>\DeclareOption</code> ...
103, 160, 213	1, 5, 6, 7, 8, 9, 10, 11, 16
<code>\cs_if_exist:NTF</code> ...	<code>\DeclareRobustCommand</code>
3, 24, 27, 199, 422	... 4, 13, 31, 38, 43, 48, 49, 53, 83, 130
<code>\cs_if_exist_p:N</code>	<code>\DeclareSymbolFont</code>
30	16, 74
<code>\cs_new:Nn</code>	<code>\DeclareSymbolFontAlphabet</code>
1, 1, 1, 4, 5, 5, 7, 7, 10, 11, 12, 13, 15,	76
15, 19, 25, 39, 40, 41, 43, 44, 47, 52,	<code>\DeclareTextCommand</code>
52, 61, 67, 67, 70, 72, 73, 74, 79, 79,	7, 13, 19
94, 94, 99, 112, 116, 118, 121, 132,	<code>\DeclareTextComposite</code>
144, 146, 147, 151, 156, 157, 158,	31
159, 160, 161, 168, 168, 169, 170,	<code>\DeclareTextCompositeCommand</code>
172, 175, 176, 185, 189, 196, 201,	37
220, 246, 252, 257, 259, 267, 282,	<code>\DeclareTextFontCommand</code>
282, 303, 303, 314, 319, 325, 325,	9, 110, 151
329, 339, 359, 373, 397, 405, 421,	<code>\DeclareTextSymbol</code>
436, 477, 496, 501, 509, 515, 525,	25
537, 599, 600, 605, 618, 628, 659, 688	<code>\DeclareUnicodeAccent</code>
<code>\cs_new:Npn</code>	6
1, 2, 3,	<code>\DeclareUnicodeEncoding</code>
4, 5, 6, 7, 8, 9, 10, 57, 65, 66, 73, 197, 238	21, 39
<code>\cs_new_eq:NN</code>	<code>\def</code>
56	28, 35, 43
<code>\cs_new_protected:Nn</code>	<code>\defaultfontfeatures</code>
..... 1, 2, 8, 285, 292, 670, 715	73
<code>\cs_new_protected:Npn</code>	<code>\Delta</code>
..... 33, 45, 58, 61, 76, 83, 114	53
<code>\cs_set:Npn</code>	dim commands:
1, 5, 9,	<code>\dim_compare:nNnTF</code>
31, 32, 33, 34, 49, 55, 61, 65, 67, 79,	328
80, 81, 82, 111, 112, 113, 152, 153,	<code>\dim_eval:n</code>
174, 347, 439, 638, 675, 681, 694, 700	3, 7
<code>\cs_set_eq:NN</code>	<code>\dim_new:N</code>
19, 43, 60, 63,	42, 43, 44
80, 85, 86, 87, 87, 88, 89, 90, 91, 170, 180	<code>\dim_set:Nn</code>
<code>\cs_to_str:N</code> 101, 105, 106, 127, 162, 215	327
<code>\cs_undefine:N</code>	<code>\dim_to_fp:n</code>
86, 242, 520	316, 317
<code>\cyrillicencoding</code>	dim internal commands:
51, 55, 81	<code>__dim_eval:w</code>
	76
	<code>__dim_eval_end:</code>
	76
	<code>\do</code>
	39
	<code>\dospecials</code>
	39
	<code>\dot</code>
	26
	<code>\DTX</code> ...
	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
	14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
	E
	<code>\else</code>
	17, 37, 95, 202, 709, 710
	else commands:
	<code>\else:</code>
	48
	<code>\em</code>
	83, 110

D

<code>\date</code>	56
<code>\ddot</code>	20

\emfontdeclare	58, 162
\emminnershape	110, 162
\emph	110
\emreset	74, 106, 110
\emshape	110, 162
\EncodingAccent	15
\EncodingCommand	9
\EncodingComposite	27
\EncodingCompositeCommand	33
\encodingdefault	21, 39, 57, 79, 307
\EncodingSymbol	21
\endinput	8, 13, 22, 29
environments:	
listingcont*	61
verbatim*	45
\ExecuteOptions	21
exp commands:	
\exp_after:wN	441
\exp_args:NNNx	321
\exp_args:Nnnx	193
\exp_args:Nnx	43, 44, 45, 49, 50
\exp_args:No	17, 60, 89, 290, 336, 410, 602, 612
\exp_args:Nv	281
\exp_args:Nx	64, 428
\exp_args:Nxx	185
\exp_last_unbraced:Nn	31, 32, 33, 34
\exp_not:N	15, 16, 17, 20, 33, 34, 35, 51, 52, 53, 83, 104, 106, 107, 108, 238, 239, 242, 243, 251, 252, 276, 277, 407, 542, 554, 567, 568
\exp_not:n	13, 31, 49, 54, 60, 191, 260, 541, 553
\expandafter	29
F	
\familydefault	20, 38, 56
\FancyVerbSpace	80
\fi	19, 29, 32, 37, 97, 98, 110, 116, 204, 358, 367, 709, 710
fi commands:	
\fi:	50
file commands:	
\file_if_exist:nTF	25, 89
\file_input:n	90
\filedate	56
\fileversion	56
\fmtname	29
\font	3, 4, 7, 9, 12, 27, 38, 42, 50, 64, 67, 77, 80, 90, 97, 103, 107, 110, 120, 135, 160, 213, 311, 332, 353, 354, 355, 361, 362, 363, 374, 379, 384, 401, 413
font commands:	
\l_fontspec_font	42, 149, 154, 160, 167
\fontdimen	80, 81, 327, 353, 354, 355, 361, 362, 363, 374, 379, 384
\fontdimen8	80
\fontencoding	4, 15, 33, 51, 107, 307
\fontfamily	16, 29, 34, 52, 106, 161, 308
\fontname	40, 56
\fontshape	7, 22, 23
\fontspec	1, 24, 31, 42, 126
fontspec commands:	
\fontspec_calc_scale:n	79
\fontspec_complete_fontname:Nn	128, 138, 143, 158, 179, 183, 193, 259, 316, 319
\g_fontspec_default_fontopts_tl	31
\g_fontspec_encoding_tl	11, 41, 42, 44, 48, 49, 51, 52, 55, 56, 74, 75, 77, 78, 78, 79, 80, 81, 84, 87, 88, 89, 91, 92, 650
\l_fontspec_family_tl	41, 42, 77, 153, 166
\l_fontspec_feature_string_tl	19, 53
\fontspec_font_if_exist:n	171
\fontspec_font_if_exist:nTF	180
\l_fontspec_fontname_tl	8, 18, 21, 42, 45, 79, 102, 115, 120, 125, 126, 130, 140, 197, 274, 288, 293, 298, 305, 308, 401, 415, 418, 423, 426, 447, 468, 479, 492, 511, 560, 568, 582, 590
\fontspec_gset_family:Nnn	63, 64, 69, 70, 75, 76, 81, 82, 156
\fontspec_gset_fontface:NNnn	169
\fontspec_if_aat_feature:nn	5
\fontspec_if_aat_feature:nnTF	5
\fontspec_if_current_feature:n	181
\fontspec_if_current_feature:nTF	181, 281
\fontspec_if_current_language:n	131
\fontspec_if_current_language:nTF	131
\fontspec_if_current_script:n	116
\fontspec_if_current_script:nTF	116
\fontspec_if_feature:n	34
\fontspec_if_feature:nnn	60
\fontspec_if_feature:nnnTF	60
\fontspec_if_feature:nTF	34
\fontspec_if_fontspec_font:	1

\fontspec_if_fontspec_font:TF . 1 , 7 , 25 , 36 , 62 , 75 , 88 , 105 , 118 , 133 , 147	\grave 19
\fontspec_if_language:n 86	group commands:
\fontspec_if_language:nn 103	\group_begin:
\fontspec_if_language:nnTF 103 4 , 15 , 28 , 37 , 51 , 72 , 119 , 149 , 173 , 266 , 287 , 294 , 305 , 390 , 399 , 518
\fontspec_if_language:nTF 86	\group_end: 21 , 33 , 39 , 41 , 42 , 81 , 128 , 160 , 177 , 178 , 274 , 290 , 297 , 322 , 394 , 395 , 403 , 521
\fontspec_if_opentype: 23	H
\fontspec_if_opentype:TF 23	\hat 25 , 114
\fontspec_if_script:n 73	\hbar 71
\fontspec_if_script:nTF 73	\hbox 37
\fontspec_if_small_caps: 189	I
\fontspec_if_small_caps:TF 189	\IfBooleanTF 118 , 130
\fontspec_maybe_setup_maths: 94	\ifcase 348
\fontspec_merge_shape:n	\IfFontExistsTF 180
. 19 , 41 , 46 , 51 , 56	\IfFontFeatureActiveTF 124 , 257
\fontspec_new_lang:nn 118 , 329	\ifmmode 37
\fontspec_new_script:nn 114 , 282	\IfNoValueTF 75
\fontspec_parse_colour:niii . 431 , 439	\ifnum 92 , 198 , 355
\fontspec_parse_cv:w 238 , 251	\ifx 15 , 29 , 30 , 98 , 110 , 116 , 709 , 710
_fontspec_parse_wordspace:w 344 , 347	\ignorespaces 4 , 9 , 14 , 19 , 78 , 166
\fontspec_print_visible_spaces:	\InputIfFileExists 3
. 15 , 43 , 51 , 57 , 72	int commands:
\fontspec_select:nn 43	\int_case:nn 57 , 78 , 83
\fontspec_set_em_level:n 159	\int_case:nnTF 333
\fontspec_set_family:Nnn	\int_compare:nNnTF 146 , 717
. 3 , 9 , 27 , 45 , 101 , 146	\int_compare:nTF 32 , 53 , 74 , 100 , 274 , 427 , 430 , 461 , 683 , 702
\fontspec_set_fontface:NNnn 159	\int_compare_p:nNn 90 , 141 , 196
\fontspec_set_strong_level:n 160	\int_eval:n 251
\fontspec_setup_maths: 1 , 134	\int_if_even:nTF 37
\fontspec_tmp: 60 , 63	\int_incr:N 79 , 96 , 100 , 126 , 141 , 152 , 203
\fontspec_visible_space: 2 , 19 , 80 , 87	\int_new:N
\fontspec_visible_space_fallback: 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39
. 6 , 8	\int_set:Nn 11 , 42 , 45 , 76 , 84 , 87 , 94 , 94 , 96 , 137 , 137 , 143 , 149 , 159 , 160 , 188 , 201 , 295 , 339 , 452 , 653 , 705
fontspec internal commands:	\int_to_hex:n 462
\l_fontspec_check_bool	\int_use:N
. 89 , 93 , 99 , 108 , 140 , 148 , 155 , 164 78 , 93 , 97 , 102 , 125 , 134 , 138 , 143
_fontspec_update_featstr:n 97	\int_zero:N 61 , 88 , 105 , 117 , 139 , 146 , 156 , 157 , 194 , 356 , 654 , 655 , 656 , 719
\FONTSPECCTX 2	\l_tmpa_int 88 , 90 , 92 , 94 , 96 , 194 , 196 , 199 , 201 , 203
\FontspecSetCheckBoolFalse 65	\l_tmpb_int 87 , 90 , 94 , 188 , 196 , 201
\FontspecSetCheckBoolTrue 65	\itdefault 2 , 11 , 13 , 15 , 41 , 77 , 84 , 89 , 307 , 309 , 529 , 530 , 534 , 546 , 548
fp commands:	\itscdefault 2 , 7 , 11 , 13 , 15 , 16 , 17 , 558 , 559
\fp_eval:n 300 , 316	\itshape 38 , 66 , 112
\fp_new:N 40 , 41	
G	
\g 116	
\Gamma 52	
\gdef 2	
\GetFileInfo 55	
\global 7 , 69	

K		
keys commands:		
\l_keys_choice_int . . .	53, 57, 74, 78, 83	
\keys_define:nn	3,	
	3, 7, 9, 20, 20, 22, 28, 48, 69, 69, 81,	
	92, 170, 203, 212, 214, 222, 230, 237,	
	244, 245, 258, 268, 277, 284, 324,	
	331, 416, 520, 528, 537, 547, 552, 574	
\keys_if_choice_exist:nnnTF .	180, 190	
\keys_if_exist:nnTF		
	177, 187, 208, 226, 234, 241	
\l_keys_key_tl	114, 119, 124, 129	
\keys_set:nn	8, 13, 32,	
	42, 85, 106, 111, 184, 185, 200, 205,	
	210, 213, 231, 238, 245, 318, 348, 482	
\keys_set_groups:nnn	411	
\keys_set_known:nn	15	
\keys_set_known:nnN . .	70, 86, 150, 441	
\l_keys_value_tl . . .	114, 119, 124, 129	
L		
\l	31, 51, 54, 64, 65, 66, 71	
\Lambda	55	
\latinencoding	52, 56, 82	
\leavevmode	37	
\let	39	
\liningnums	2, 35, 135, 285	
listingcont*(environment)	61	
lua commands:		
\lua_now:e	6, 79, 80, 81, 82, 107	
\LuaLaTeX	34	
M		
\mathalpha		
	18, 19, 20, 21, 22, 23, 24, 25, 26, 27,	
	42, 43, 44, 45, 46, 47, 48, 49, 50, 51,	
	52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62	
\mathbf	56, 78, 88, 116	
\mathbin	63	
\mathchar	71	
\mathchardef	29	
\mathclose	36, 39, 66, 68	
\mathdollar	70	
\mathit	40, 56, 77, 84, 89	
\mathopen	65, 67	
\mathord	69, 70	
\mathpunct	31, 38	
\mathrel	37, 64	
\mathring	27	
\mathrm	28, 76, 87, 116	
\mathsf	79, 91	
\mathtt	80, 92	
		\mddefault 74, 75, 77, 79, 80, 87, 89, 305,
		307, 308, 544, 546, 547, 556, 558, 560
		\mkern 71
	msg commands:	
	\msg_error:nn	1
	\msg_error:nnn	2, 3
	\msg_fatal:nn	21
	\msg_info:nn	7
	\msg_info:nnn	8
	\msg_info:nnnn	9
	\msg_line_context:	96
	\msg_new:nnn	11, 14, 15
	\msg_new:nnnn	12, 16
	\msg_redirect_module:nnn 13, 14, 18, 19	
	\msg_redirect_name:nnn	481
	\msg_trace:nn	10
	\msg_warning:nn	4
	\msg_warning:nnn	5
	\msg_warning:nnnn	6
N		
\newAATfeature	92, 175	
\NewDocumentCommand	1, 6, 11, 16,	
	21, 25, 29, 33, 37, 41, 43, 45, 49, 53,	
	57, 59, 61, 65, 69, 73, 80, 84, 88, 92,	
	96, 100, 104, 108, 112, 116, 120, 124, 135	
\newfontface	30, 57	
\newfontfamily	41	
\newfontfeature	32, 32, 32, 88, 168	
\newfontlanguage	1,	
	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,	
	15, 16, 17, 18, 19, 20, 21, 22, 23, 24,	
	25, 26, 27, 28, 29, 30, 31, 32, 33, 34,	
	35, 36, 37, 38, 39, 40, 41, 42, 43, 44,	
	45, 46, 47, 48, 49, 50, 51, 52, 53, 54,	
	55, 56, 57, 58, 59, 60, 61, 62, 63, 64,	
	65, 66, 67, 68, 69, 70, 71, 72, 73, 74,	
	75, 76, 77, 78, 79, 80, 81, 82, 83, 84,	
	85, 86, 87, 88, 89, 90, 91, 92, 93, 94,	
	95, 96, 97, 98, 99, 100, 101, 102, 103,	
	104, 105, 106, 107, 108, 109, 110,	
	111, 112, 113, 114, 115, 116, 116,	
	117, 118, 119, 120, 121, 122, 123,	
	124, 125, 126, 127, 128, 129, 130,	
	131, 132, 133, 134, 135, 136, 137,	
	138, 139, 140, 141, 142, 143, 144,	
	145, 146, 147, 148, 149, 150, 151,	
	152, 153, 154, 155, 156, 157, 158,	
	159, 160, 161, 162, 163, 164, 165,	
	166, 167, 168, 169, 170, 171, 172,	
	173, 174, 175, 176, 177, 178, 179,	

180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383	
<code>\newfontscript</code>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, <u>112</u> , 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145
<code>\newICUfeature</code>	<u>100</u>
<code>\newopentypefeature</code>	96, <u>185</u>
<code>\normalfont</code>	23, 41, 59, 73, <u>120</u> , <u>155</u>
<code>\normalsize</code>	50, <u>519</u>
<code>\null</code>	37
<code>\nullfont</code>	15
<code>\numexpr</code>	44
O	
<code>\oldstylenums</code>	2, <u>35</u> , <u>128</u> , <u>285</u>
<code>\Omega</code>	62
<code>\or</code>	351, 359, 363
P	
<code>\par</code>	70
<code>Path</code>	<u>24</u>
<code>\Phi</code>	60
<code>\Pi</code>	57
<code>\postexhyphenchar</code>	656
<code>\posthyphenchar</code>	654
<code>\preexhyphenchar</code>	655
<code>\prehyphenchar</code>	653
prg commands:	
<code>\prg_new_conditional:Nnn</code> 1, 5, 23, 25, 26, 34, 44, 58, 60, 73, 75, 86, 103, 116, 121, 125, 131, 169, 171, 174, 181, 189, 220, 263, 388
<code>\prg_return_false:</code>	3, 13, 16, 18, 20, 24, 28, 29, 30, 31, 36, 41, 49, 50, 51, 53, 57, 67, 68, 69, 71, 80, 82, 82, 84, 97, 99, 99, 101, 110, 112, 114, 115, 123, 125, 127, 129, 132, 140, 142, 144, 155, 164, 172, 178, 180, 187, 204, 206, 207, 217, 231, 279, 282, 394
<code>\prg_return_true:</code>	3, 13, 16, 24, 28, 29, 36, 42, 47, 50, 54, 67, 68, 79, 80, 97, 99, 110, 111, 123, 125, 129, 140, 155, 164, 172, 177, 177, 187, 205, 206, 217, 237, 243, 282, 395
<code>\prg_set_conditional:Nnn</code>	13, 21, 26, 35
prop commands:	
<code>\prop_gc clear:N</code>	60, 116
<code>\prop_get:NnN</code>	41, 44, 47, 48, 93, 95, 123, 138, 151, 152
<code>\prop_get:NnNTF</code>	91, 92, 94, 102, 102, 105, 132, 135, 143, 228, 248, 409
<code>\prop_gput:Nnn</code>	11, 78, 90, 125, 135, 140, 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, 254, 255, 256, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 270, 271, 271, 272, 272, 273, 274, 275, 276,

277, 277, 278, 278, 279, 279, 280, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343 \prop_gput_if_new:Nnn 77, 89, 124 \prop_gremove:Nn 139 \prop_if_in:NnTF 9, 87 \prop_map_inline:Nn 312 \prop_new:N 68, 69, 70, 71, 72, 73, 74, 75, 76, 269 \prop_put:Nnn 87, 88, 171, 204, 335 \providecommand 1, 2, 2, 3, 3, 4, 5, 6 \ProvideDocumentCommand 55, 71 \providefontface 69 \providefontfamily 53 \ProvidesExplFile 49 \ProvidesExplPackage 44, 45, 46 \Psi 61	\selectfont 5, 7, 17, 22, 23, 35, 53, 108, 161, 309 seq commands: \seq_if_empty:NTF 160 \seq_new:N 45 \seq_put_right:Nn 148, 163 \setboldmathrm 25, 28, 67, 93, 116 \setfontface 65 \setfontfamily 49 \setmainfont 6, 7, 24, 28, 114 \SetMathAlphabet 77, 78, 79, 80, 84, 87, 88, 89, 91, 92 \setmathrm 21, 61, 92, 116 \setmathsf 29, 73, 94 \setmathtt 33, 79, 95 \setmonofont 16, 43 \setromanfont 37 \setsansfont 11, 25 \SetSymbolFont 17, 75, 81 \settoheight 330 \sfdefault 28, 38, 46, 97, 122 \sffamily 31 \Sigma 58 \sishape 2 \sldefault 3, 12, 14, 16, 46, 308, 310, 530, 533, 547, 549 \slscdefault . . 3, 12, 14, 15, 16, 18, 560, 561 \slshape 38, 64 \space 34, 39, 44, 197 str commands: \c_backslash_str 87 \c_colon_str 239, 252 \str_case:nn 74, 542, 554 \str_case:nnTF 199, 290 \str_case_e:nnTF 369 \str_if_eq:eeTF 20, 38, 56, 72, 116, 156, 216, 393 \str_if_eq:nnTF 93, 124, 139, 332, 398, 478 \str_if_eq_p:ee 305, 313, 314, 315, 529, 530 \str_if_eq_p:nn 677, 696 \str_lower_case:n 72, 116 \string 21, 56, 76, 96 \strong 151 \strongenv 130, 151 \strongfontdeclare 114, 161 \strongreset 121, 147, 151 \suppressfontnotfounderror 11
Q	
\quad 56 quark commands: \q_nil 72, 73, 196, 197, 239, 252, 672, 675, 678, 679, 681, 691, 694, 697, 698, 700 \q_stop 344, 347	
R	
\relax . . . 5, 6, 29, 37, 44, 45, 50, 55, 69, 355 \renewcommand 71 \RenewDocumentCommand 47, 63, 130 \renewfontface 61 \renewfontfamily 45 \RequirePackage 5, 43, 48, 48, 62 \RequirePackageWithOptions 7, 12 \rmdefault . . 10, 20, 29, 45, 96, 116, 121, 308 \rmfamily 13, 79, 332	
S	
scan commands: \scan_stop: 3, 5, 7, 46, 54 \scdefault 2, 3, 11, 12, 13, 14, 17, 18, 51, 511, 512, 513, 556, 557 \scshape 38 \select 19	

141, 142, 143, 144, 145, 146, 146, 147, 148, 152, 153, 154, 155, 156, 159	\UndeclareComposite 82
\tl_put_left:Nn 46	\UndeclareSymbol 64
\tl_put_right:Nn 134, 390, 400, 412, 489	\UndeclareTextCommand 68, 74, 80
\tl_remove_all:Nn ... 47, 86, 235, 323	\unexpanded 96, 123
\tl_remove_once:Nn 52	\UnicodeEncodingName 13, 19, 25, 31, 37, 57, 58, 62, 68, 74, 80, 87
\tl_replace_all:Nnn 90, 99, 322	\UnicodeFontFile 1, 47, 49, 51, 53, 55
\tl_set:Nn 21, 22, 26, 28, 29, 34, 39, 39, 42, 45, 46, 46, 47, 53, 53, 55, 58, 76, 81, 85, 86, 99, 100, 104, 105, 107, 112, 118, 119, 123, 124, 127, 136, 139, 149, 150, 154, 160, 161, 162, 163, 164, 167, 196, 214, 215, 234, 239, 243, 250, 253, 255, 267, 293, 294, 295, 299, 300, 314, 321, 323, 327, 338, 351, 354, 355, 359, 362, 366, 373, 378, 382, 383, 407, 408, 428, 441, 447, 459, 468, 472, 486, 500, 517, 558, 572, 580, 652	\UnicodeFontName 2
\tl_set_eq:NN 10, 21, 28, 39, 41, 46, 49, 51, 52, 55, 56, 57, 57, 62, 126, 157, 168, 170, 193, 280, 426, 439, 560, 568, 582, 590, 663, 664, 665	\UnicodeFontTeXLigatures 3, 4, 47, 49, 51, 53, 55
\tl_to_str:N 21	\updefault . 17, 18, 56, 74, 75, 78, 79, 80, 81, 87, 88, 91, 92, 203, 305, 306, 544, 545
\tl_to_str:n 87, 186	\upshape 38, III, III
\tl_trim_spaces:n 14, 16	\Upsilon 59
\tl_use:N 22, 33, 512	\url 40
\tmpa 28, 29	use commands:
token commands:	\use:N 99, 106
\token_to_str:N 87, 422	\use:n 11, 29, 47, 102, 155, 170, 236, 441
\ttdefault 46, 47, 56, 98, 123	\use_i:nnn 286
\ttfamily 49	\use_ii:nnn 286
\typeout 3, 5, 23, 32, 37, 54, 59, 60, 69, 71, 77, 83, 85, 86, 93, 96, 97, 106, 110, 114, 120, 123, 127, 134, 134, 136, 138, 142, 146, 154, 173, 178, 183, 183, 183, 184, 191, 199, 203, 203, 209, 209, 210, 223, 224, 228, 236, 243, 259, 260, 276, 276, 277, 279, 286, 292, 304, 341, 349, 352, 356, 360, 364, 407, 423, 444, 449, 460, 464, 479, 482, 517, 620, 624, 630, 633, 640, 690	\use_iii:nnn 272
	\use_none:nn .. 85, 86, 87, 88, 89, 90, 91
	\usefont 11
	\UTFencname 49, 80
	V
	\verb 33, 126
	\verb* 33, 125
	verbatim* (environment) 45
	\verbvisiblespace 24
	X
	\XeLaTeX 34
	\XeTeXcountvariations 355
	\XeTeXfeaturename 28
	\XeTeXfonttype 348
	\XeTeXisexclusivefeature 32
	\XeTeXOTcountfeatures 190
	\XeTeXOTcountlanguages 138
	\XeTeXOTcountscripts 87
	\XeTeXOTfeaturetag 198
	\XeTeXOTlanguagetag 144
	\XeTeXOTscripttag 92
	\XeTeXpicfile 60, 61, 63
	\XeTeXselectorname 34, 39, 44
	\Xi 56
U	
\UndeclareAccent 64	
\UndeclareCommand 64	