

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/>

2019/01/24 v2.7

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	59
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	71
1	Graphite/AAT code	71

XIII	fontspec-code-keyval.dtx	73
1	Font loading (keyval) definitions	73
1.1	Pre-pre-parsing stages	73
1.2	Pre-parsed features	75
1.3	Font faces	75
1.4	General font-independent features	78
XIV	fontspec-code-feat-opentype.dtx	88
1	OpenType feature definitions	88
2	Regular key=val / tag definitions	88
2.1	Ligatures	88
2.2	Letters	88
2.3	Numbers	89
2.4	Vertical position	89
2.5	Contextuals	90
2.6	Diacritics	90
2.7	Kerning	90
2.8	Fractions	90
2.9	Style	91
2.10	CJK shape	91
2.11	Character width	92
2.12	Vertical	92
3	OpenType features that need numbering	92
3.1	Alternate	92
3.2	Variant / StylisticSet	93
3.3	CharacterVariant	93
3.4	Annotation	94
3.5	Ornament	94
4	Script and Language	94
4.1	Script	94
4.2	Language	95
5	Backwards compatibility	96
XV	fontspec-code-scripts.dtx	97
1	Font script definitions	97
XVI	fontspec-code-lang.dtx	100
1	Font language definitions	100

XVII	fontspec-code-feat-aat.dtx	108
1	AAT feature definitions	108
1.1	Ligatures	108
1.2	Letters	108
1.3	Numbers	109
1.4	Contextuals	109
1.5	Diacritics	109
1.6	Vertical position	109
1.7	Fractions	109
1.8	Alternate	109
1.9	Variant / StylisticSet	110
1.10	Style	110
1.11	CJK shape	110
1.12	Character width	111
1.13	Annotation	111
XVIII	fontspec-code-enc.dtx	112
1	Extended font encodings	112
XIX	fontspec-code-math.dtx	115
1	Selecting maths fonts	115
XX	fontspec-code-closing.dtx	120
1	Closing code	120
1.1	Finishing up	120
XXI	fontspec-code-xfss.dtx	121
1	Changes to the NFSS	121
1.1	Italic small caps and so on	121
1.2	Emphasis	122
1.3	Strong emphasis	124
XXII	fontspec-code-patches.dtx	126
1	Patching code	126
1.1	\-	126
1.2	Verbatims	126
	Index	129

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 {2019/01/24}{2.7}{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   = "2.7",
63 <lua>  date      = "2019/01/24",
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   \directlua{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_@@_graphite_bool
11 \bool_new:N \l_@@_fontcfg_bool
12 \bool_set_true:N \l_@@_fontcfg_bool
```

For dealing with legacy maths:

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

For package options:

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

```

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

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

84 \tl_new:N \l_fontspec_mode_tl
85 \tl_new:N \g_@@_curr_series_tl
86 \tl_new:N \g_@@_defined_shapes_tl
87 \tl_new:N \g_@@_nfss_enc_tl
88 \tl_new:N \g_@@_nfss_family_tl
89 \tl_new:N \g_@@_single_feat_tl
90 \tl_new:N \l_@@_basename_tl
91 \tl_new:N \l_@@_curr_fontname_tl
92 \tl_new:N \l_@@_curr_bfname_tl
93 \tl_new:N \l_@@_ext_filename_tl
94 \tl_new:N \l_@@_extension_tl
95 \tl_new:N \l_@@_font_path_tl
96 \tl_new:N \l_@@_fontid_tl
97 \tl_new:N \l_@@_fontname_tl
98 \tl_new:N \l_@@_hexcol_tl
99 \tl_new:N \l_@@_nfss_sc_tl
100 \tl_new:N \l_@@_nfss_tl
101 \tl_new:N \l_@@_nfss_fam_tl
102 \tl_new:N \l_@@_opacity_tl
103 \tl_new:N \l_@@_optical_size_tl
104 \tl_new:N \l_@@_options_tl
105 \tl_new:N \l_@@_saved_fontname_tl
106 \tl_new:N \l_@@_scale_tl

```

```

107 \tl_new:N \l_@@_size_tl
108 \tl_new:N \l_@@_sizedfont_tl
109 \tl_new:N \l_@@_this_font_tl
110 \tl_new:N \l_@@_tmp_tl
111 \tl_new:N \l_@@_tmpa_tl
112 \tl_new:N \l_@@_tmpb_tl
113 \tl_new:N \l_@@_ttc_index_tl
114 \tl_new:N \l_@@_emshape_query_tl
115 \tl_new:N \l_@@_em_switch_tl
116 \tl_new:N \l_@@_em_tmp_tl
117 \tl_new:N \l_@@_strong_tmp_tl
118 \tl_new:N \l_@@_strong_switch_tl
119 \tl_new:N \l_@@_hyphenchar_tl
120 \tl_new:N \l_@@_smcp_shape_tl

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

    Defaults:

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

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

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

138 \tl_new:N \l_@@_script_name_tl
139 \tl_set:Nn \l_@@_script_name_tl {CustomDefault}
140 \tl_new:N \l_fontspeg_script_tl
141 \tl_new:N \l_@@_lang_name_tl
142 \tl_set:Nn \l_@@_lang_name_tl {Default}
143 \tl_new:N \l_fontspeg_lang_tl

144 \tl_new:N \l_@@_mapping_tl
145 \tl_new:N \l_@@_punctspace_adjust_tl
146 \tl_new:N \l_@@_wordspace_adjust_tl
147 \tl_new:N \l_@@_postadjust_tl

148 \tl_const:Nn \c_@@_hexcol_tl {000000}
149 \tl_const:Nn \c_@@_opacity_tl {FF~}
150 \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.

151 \tl_new:N \g_@@_rawfeatures_sclist
152 \tl_new:N \l_@@_pre_feat_sclist

```

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

```
153 \tl_new:N \l_@@_rmfamily_family_tl  
154 \tl_new:N \l_@@_sffamily_family_tl  
155 \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} {no-mapping}
152 {
153   Input mapping not (yet?) supported in LuaTeX.
154 }
155 \@@_msg_new:nnn {fontspec} {no-mapping-ligtext}
156 {
157   Input mapping not (yet?) supported in LuaTeX.\\
158   Use "Ligatures=TeX" instead of "Mapping=tex-text".
159 }
160 \@@_msg_new:nnn {fontspec} {cm-default-obsolete}
161 {
162   The "cm-default" package option is obsolete.
163 }
164 \@@_msg_new:nnn {fontspec} {fakebold-only-xetex}
165 {
166   The "FakeBold" and "AutoFakeBold" options are only available with XeLaTeX.\\
167   Option ignored.
168 }
169 \@@_msg_new:nnn {fontspec} {font-index-needs-ttc}
170 {
171   The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
172   Feature ignored.
173 }
174 \@@_msg_new:nnn {fontspec} {feat-cannot-remove}
175 {
176   The "#1" feature cannot be deactivated. Request ignored.
177 }

```

1.3 Info messages

```

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

```



```

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

    Reset 'space' behaviour:
220 \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` These strange set functions are to simplify returning code from LuaTeX:

```

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

```

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

`\@@_keys_set_known:nnN`

```

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

```

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

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

```

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

```

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

1.4 expl3 variants

```

78 \cs_generate_variant:Nn \int_set:Nn {Nv}
79 \cs_generate_variant:Nn \keys_set:nn {nx}
80 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
81 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
82 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
83 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
84 \cs_generate_variant:Nn \prop_gput:Nnn {NxN}
85 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
86 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
87 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
88 \cs_generate_variant:Nn \tl_if_empty_p:n {e}
89 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
90 \cs_generate_variant:Nn \tl_if_empty:nF {x}
91 \cs_generate_variant:Nn \tl_if_empty:nF {f}
92 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
93 \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 \newfontface { m O{} m O{} }
54 {
55   \@@_main_newfontface:nnn {#1} {#2,#4} {#3}
56 }

```

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

```

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

```

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

```

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

```



```

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

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

```

`\newICUfeature` Deprecated.

```

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

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

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

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

```

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

```

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

```

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

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

```

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

```

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

```

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

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

```

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

```
112 \AtBeginDocument
113 {
114   \RenewDocumentCommand \oldstylenums {m}
115   {
116     \@@_main_oldstylenums:n {#1}
117   }
118 }
```

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

`\liningnums`

```
119 \NewDocumentCommand \liningnums {m}
120 {
121   \@@_main_liningnums:n {#1}
122 }
```

(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_set_family:Nnn \g_@@_mathrm_tl {#1} {#2}
64     \fontspec_set_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
65     \@@_setmathrm_hook:nn {#1} {#2}
66 }

```

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

`\setboldmathrm`

```

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

```

```

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

```

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

`\setmathsf`

```

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

```

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

`\setmathtt`

```

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

```

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

Hooks:

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

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

1.2 Font feature selection

\@@_set_default_features:nn

```

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

```

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

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

```

121 \cs_new:Nn \@@_set_font_default_features:nnn
122 {
123     <debug> \typeout{\unexpanded{\set_font_default_features:nnn:{#1}{#2}{#3}}}
124     \clist_map_inline:nn {#2}
125     {
126         \tl_if_single:nTF {##1}
127         { \tl_set: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 }

```

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

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

```

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

```

Defaults:

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

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

1.4 High level conditionals

\IfFontFeatureActiveTF

```

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

```

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

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_fontspec_script_tl
48             \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag} \l_fontspec_lang_tl
49
50             \@@_check_ot_feat:NnTF \font {#1} {\prg_return_true:} {\prg_return_false:}
51         }
52         {
53             \prg_return_false:
54         }
55     }
56     {
57         \prg_return_false:
58     }
59 }

```

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

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

```

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

```

```

67         \@@_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_fontspec_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 $\langle features \rangle$ and $\langle font \rangle$, and stores the name in the variable $\langle family \rangle$. See the standard fontspec user commands for applications of this function.

We want to store the actual name of the font family within the $\langle family \rangle$ 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 \fontspec_set_family:Nnn
147 {
148     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
149     \tl_set:Nn \l_@@_family_label_tl {#1}
150     \@@_select_font_family:nn {#2} {#3}
151     \tl_if_exist:NF #1 { \tl_new:N #1 }
152     \tl_set_eq:NN #1 \l_fontspec_family_tl
153     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
154 }

```

```

155 \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 fontdi-
                             mens will be lost. (Discovered in unicode-math.) Investigate!

```

```

156 \tl_new:N \l_@@_fontface_cs_tl
157 \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font}
158 \cs_new:Nn \fontspec_set_fontface:NNnn
159 {
160     \tl_set:Nn \l_@@_fontface_cs_tl {#1}
161     \tl_set:Nn \l_@@_family_label_tl {#2}
162     \@@_select_font_family:nn {#3} {#4}
163     \tl_set_eq:NN #2 \l_fontspec_family_tl
164     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
165 }

```

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

```

\fontspec_font_if_exist:n

```

```

166 \prg_new_conditional:Nnn \fontspec_font_if_exist:n {TF,T,F}
167 {
168     \group_begin:
169     \@@_init:
170     \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
171     \@@_primitive_font_if_exist:nTF { \@@_construct_font_call:nn {#1} {} }
172     { \group_end: \prg_return_true: }
173     { \group_end: \prg_return_false: }
174 }

```



```
175 \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.

```
176 \prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
177 {
178   \exp_args:Nxx \tl_if_in:nnTF
179   { \@@_primitive_font_current_name: } { \tl_to_str:n {#1} }
180   { \prg_return_true: } { \prg_return_false: }
181 }
```

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

`\fontspec_if_small_caps:TF`

```
182 \prg_new_conditional:Nnn \fontspec_if_small_caps: {TF,T,F}
183 {
184   \@@_if_merge_shape:nTF {sc}
185   {
186     \tl_set_eq:Nc \l_@@_smcp_shape_tl { \@@_shape_merge:nn {f@shape} {sc} }
187   }
188   {
189     \tl_set:Nn \l_@@_smcp_shape_tl {sc}
190   }
191
192   \cs_if_exist:cTF { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
193   {
194     \tl_if_eq:ccTF
195     { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
196     { \f@encoding/\f@family/\f@series/\updefault }
197     { \prg_return_false: }
198     { \prg_return_true: }
199   }
200   { \prg_return_false: }
201 }
```

(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_fontspeg_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:nn` 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_fontspeg_script
280   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_fontspeg_lang_t
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_fontspec_renderers_tl { \tl_set:Nn \l_fontspec_renderers_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_fontspec_renderers_tl { \tl_set:Nn \l_fontspec_renderers_tl {/OT} }
363 \or
364 <debug> \typeout{::: Graphite}
365 \bool_set_true:N \l_@@_graphite_bool
366 \tl_if_empty:NT \l_fontspec_renderers_tl { \tl_set:Nn \l_fontspec_renderers_tl {/GR} }
367 \fi
368 </XE>

```

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

LuaTeX only supports one:

```

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

```

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

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

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

We can't match external fonts in this way (in X_YTeX anyway; todo: test with LuaTeX). If `` is not empty, then it's already been specified by the user so abort. If `<Base font name>` is not given, we also abort for obvious reasons.

If `` is empty, then proceed. If not found, `` remains empty. Otherwise, we have a match.

```

373 \cs_new:Nn \@@_set_autofont:Nnn
374 {
375   \bool_if:NF \l_@@_external_bool
376   {
377     \tl_if_empty:xF {#2}
378     {
379       \tl_if_empty:NT #1
380       {
381         \@@_if_autofont:nnTF {#2} {#3}
382         { \tl_set:Nx #1 {#2#3} }
383         { \@@_info:nx {no-font-shape} {#2#3} }
384       }
385     }
386   }
387 }

```

```

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:ifF {\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_fontspec_script_tl
578     {
579       script = \l_fontspec_script_tl ;
580       language = \l_fontspec_lang_tl ;
581     }

```

```

582     }
583   }
584   </XE>
585   <*LU>
586   {
587     mode      = \l_fontspec_mode_tl    ;
588     \tl_if_empty:NF \l_fontspec_script_tl
589     {
590       script   = \l_fontspec_script_tl ;
591       language = \l_fontspec_lang_tl   ;
592     }
593   }
594   </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF 595 <LU>\cs_new:Nn \@@_make_smallcaps:TF
596 <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
597 {
598   \exp_args:No \@@_check_ot_feat:NnTF \l_@@_fontface_cs_tl {smcp} {#1} {#2}
599 }
600 <*XE>
601 \cs_new:Nn \@@_make_smallcaps:TF
602 {
603   \bool_if:NTF \l_@@_ot_bool
604   { \@@_make_ot_smallcaps:TF {#1} {#2} }
605   {
606     \bool_if:NT \l_@@_atsui_bool
607     {
608       \exp_args:No \@@_make_AAT_feature_string:NnnTF
609       \l_@@_fontface_cs_tl {3} {3} {#1} {#2}
610     }
611   }
612 }
613 </XE>

```

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

```

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

```

```

\@@_remove_clashing_featstr:n 624 \cs_new:Nn \@@_remove_clashing_featstr:n

```

```

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

```

1.3 Initialisation

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

```

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

```

Executed in \@@_get_features:Nn.

```

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

```

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

```

\@@_ot_validate_tag:n 664 <*LU>

```

```

665 \cs_new_protected:Nn \@@_ot_validate_tag:n
666 {
667   \@@_ot_validate_tag:w #1 \q_nil
668 }
669 \cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
670 \cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
671 {
672   \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
673     { \@@_ot_validate_tag_aux:w #2 \c_empty_tl \c_empty_tl \q_nil }
674     { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
675 }
676 \cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
677 {
678   \int_compare:nT { \tl_count:n {#5} > 2 }
679     { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
680 }
681 </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.

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

```

682 < *XE >
683 \cs_new:Nn \@@_iv_str_to_num:Nn
684 {
685   < debug > \typeout{iv_str_to_num:~#1~/~#2}
686   \@@_strip_leading_sign:Nw #1#2 \q_nil
687 }
688 \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 then are used to reconstruct the spaces in the string to number calculation.

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

```

689 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
690 {
691   \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
692     { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
693     { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
694 }

```

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.

```

695 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
696 {
697   \int_compare:nT { \tl_count:n {#6} > 2 }
698     { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }
699 }

```

```

700     \int_set:Nn #1
701     {
702         `#2 * "10000000
703         + `#3 * "100000
704         + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
705         + \ifx \c_empty_tl #5 32 \else `#5 \fi
706     }
707 }
708 </XE>

```

```

\@@_lang_dflt_correct:N 709 <*XE>
710 \cs_new_protected:Nn \@@_lang_dflt_correct:N
711 {
712     \int_compare:nNnT {#1} = {1145457748} % "DFLT"
713     {
714         \int_zero:N #1
715     }
716 }
717 </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     \bool_set_true:N \l_@@_check_feat_bool
58
59     \tl_if_empty:nT {#1} { \bool_set_false:N \l_@@_check_feat_bool }
60     \bool_if:NT \l_@@_check_feat_bool
61     {
62         \exp_args:No \@@_check_ot_feat:NnF \l_@@_fontface_cs_tl {#1}
63         {
64             \@@_warning:nx {icu-feature-not-exist-in-font} {#1}

```

```

65         \bool_set_false:N \l_@@_proceed_bool
66     }
67 }
68
69 \bool_if:NT \l_@@_proceed_bool
70 {
71     \exp_args:Nx \@@_remove_clashing_featstr:n
72     { #2 , \@@_swap_plus_minus:n {#2} , #3 }
73
74     \@@_update_featstr:n {#2}
75 }
76 }
77 \cs_generate_variant:Nn \@@_make_OT_feature:nnn {xxx}
78 \cs_new:Nn \@@_swap_plus_minus:n { \@@_swap_plus_minus_aux:Nq #1 \q_nil }
79 \cs_new:Npn \@@_swap_plus_minus_aux:Nq #1#2 \q_nil
80 { \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.

```

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

```



```

111 { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
112 <debug>\typeout{:::~ checking::~~\l_@@_tmp_tl",~ "#2"}
113 \directlua{fontspec.check_ot_script("\l_@@_tmp_tl", "#2")}
114 \bool_if:NTF \l__fontspec_check_bool
115 {
116 <debug>\typeout{:::~ TRUE}
117 \prg_return_true:
118 }
119 {
120 <debug>\typeout{:::~ FALSE}
121 \prg_return_false:
122 }
123 </LU>
124 }
125 }
126 }

```

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

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

```

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

```

```

155         \int_set:Nn \l_@@_tmpa_int {\l_@@_tmpb_int}
156     }
157     {
158         \int_incr:N \l_@@_tmpa_int
159     }
160 }
161 \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
162 </XE>
163 <*LU>
164     \@@_ot_validate_tag:x {#2}
165     \@@_ot_validate_tag:x {#3}
166     \cs_if_eq:NNTF #1 \font
167     { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
168     { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
169     \directlua
170     {
171         fontspec.check_ot_lang( "\l_@@_tmp_tl", "#2", "#3" )
172     }
173     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
174 </LU>
175 }
176 }
177 }

```

(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'.

```

178 \prg_new_conditional:Nnn \@@_check_ot_feat:Nn {TF,F}
179 {
180     \@@_check_ot_feat:NnnnTF #1 {#2} {\l__fontspec_lang_tl} {\l__fontspec_script_tl}
181     {\prg_return_true:} {\prg_return_false:}
182 }
183 \prg_new_conditional:Nnn \@@_check_ot_feat:Nnnn {TF,F}
184 {
185     \bool_if:NTF \l_@@_never_check_bool
186     { \prg_return_true: }
187     {
188         \bool_if:NTF { \tl_if_empty_p:e {#3} || \tl_if_empty_p:e {#4} }
189         { \prg_return_false: }
190         {
191             <*XE>
192             <debug>\typeout{::~ fontspec_check_ot_feat:nnn~ {#2}{#3}{#4}}
193             \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
194             \@@_iv_str_to_num:Nx \l_@@_language_int {#3}
195             \@@_lang_dflt_correct:N \l_@@_language_int
196             \@@_iv_str_to_num:Nx \l_@@_script_int {#4}
197             \int_set:Nn \l_tmpb_int
198             {

```

```

199         \XeTeXOTcountfeatures #1
200             \l_@@_script_int
201             \l_@@_language_int
202     }
203     \int_zero:N \l_tmpa_int
204     \bool_set_false:N \l_@@_check_bool
205     \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
206     {
207         \ifnum\XeTeXOTfeaturetag #1 \l_@@_script_int \l_@@_language_int
208             \l_tmpa_int = \l_@@_strnum_int
209             \bool_set_true:N \l_@@_check_bool
210             \int_set:Nn \l_tmpa_int {\l_tmpb_int}
211         \else
212             \int_incr:N \l_tmpa_int
213         \fi
214     }
215     \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
216 </XE>
217 <*LU>
218 <debug>\typeout{::~~ fontspec_check_ot_feat:n~ {#1}}
219     \@@_ot_validate_tag:x {#2}
220     \@@_ot_validate_tag:x {#3}
221     \@@_ot_validate_tag:x {#4}
222     \cs_if_eq:NNTF #1 \font
223     { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
224     { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
225     \directlua
226     {
227         fontspec.check_ot_feat("\l_@@_tmp_tl", "#2", "#3", "#4")
228     }
229     \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
230 </LU>
231 }
232 }
233 }

```

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

1.2 OpenType feature information

```

234 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {aalt}{Access~All~Alternates}
235 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvf}{Above-base~Forms}
236 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvm}{Above-base~Mark~Positioning}
237 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvs}{Above-base~Substitutions}
238 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {afrc}{Alternative~Fractions}
239 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {akhn}{Akhands}
240 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwf}{Below-base~Forms}
241 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwm}{Below-base~Mark~Positioning}
242 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blws}{Below-base~Substitutions}
243 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {calt}{Contextual~Alternates}
244 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {case}{Case-Sensitive~Forms}

```

245 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ccmp}{Glyph-Composition~/~Decomposition}
 246 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cfar}{Conjunct-Form-After-Ro}
 247 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cجت}{Conjunct-Forms}
 248 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {clig}{Contextual-Ligatures}
 249 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cpct}{Centered-CJK-Punctuation}
 250 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {csp}{Capital-Spacing}
 251 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {csw}{Contextual-Swash}
 252 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {curs}{Cursive-Positioning}
 253 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cvNN}{Character-Variant-\$N\$}
 254 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2pc}{Petite-Capitals-From-Capitals}
 255 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2sc}{Small-Capitals-From-Capitals}
 256 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dist}{Distances}
 257 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dlig}{Discretionary-Ligatures}
 258 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dnom}{Denominators}
 259 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dtls}{Dotless-Forms}
 260 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {expt}{Expert-Forms}
 261 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {falt}{Final-Glyph-on-Line-Alternates}
 262 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin2}{Terminal-Forms-#2}
 263 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin3}{Terminal-Forms-#3}
 264 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fina}{Terminal-Forms}
 265 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {flac}{Flattened-accent-forms}
 266 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {frac}{Fractions}
 267 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fwid}{Full-Widths}
 268 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {half}{Half-Forms}
 269 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {haln}{Halant-Forms}
 270 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {halt}{Alternate-Half-Widths}
 271 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hist}{Historical-Forms}
 272 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hkna}{Horizontal-Kana-Alternates}
 273 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hlig}{Historical-Ligatures}
 274 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hngl}{Hangul}
 275 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hojo}{Hojo-Kanji-Forms}
 276 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hwid}{Half-Widths}
 277 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {init}{Initial-Forms}
 278 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {isol}{Isolated-Forms}
 279 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ital}{Italics}
 280 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jalt}{Justification-Alternates}
 281 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp78}{JIS78-Forms}
 282 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp83}{JIS83-Forms}
 283 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp90}{JIS90-Forms}
 284 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp04}{JIS2004-Forms}
 285 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {kern}{Kerning}
 286 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lfb}{Left-Bounds}
 287 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {liga}{Standard-Ligatures}
 288 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ljmo}{Leading-Jamo-Forms}
 289 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lnum}{Lining-Figures}
 290 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {locl}{Localized-Forms}
 291 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltra}{Left-to-right-alternates}
 292 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltrm}{Left-to-right-mirrored-forms}
 293 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mark}{Mark-Positioning}
 294 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {med2}{Medial-Forms-#2}
 295 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {medi}{Medial-Forms}

296 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mgrk}{Mathematical-Greek}
 297 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mkmk}{Mark-to-Mark-Positioning}
 298 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mset}{Mark-Positioning-via-Substitution}
 299 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nalt}{Alternate-Annotation-Forms}
 300 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nlck}{NLC-Kanji-Forms}
 301 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nukt}{Nukta-Forms}
 302 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {numr}{Numerators}
 303 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {onum}{Oldstyle-Figures}
 304 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {opbd}{Optical-Bounds}
 305 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ordn}{Ordinals}
 306 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ornm}{Ornaments}
 307 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {palt}{Proportional-Alternate-Widths}
 308 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pcap}{Petite-Capitals}
 309 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pkna}{Proportional-Kana}
 310 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pnum}{Proportional-Figures}
 311 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pref}{Pre-Base-Forms}
 312 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pres}{Pre-base-Substitutions}
 313 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstf}{Post-base-Forms}
 314 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {psts}{Post-base-Substitutions}
 315 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pwid}{Proportional-Widths}
 316 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {qwid}{Quarter-Widths}
 317 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rand}{Randomize}
 318 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rclt}{Required-Contextual-Alternates}
 319 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rkrf}{Rakar-Forms}
 320 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rlig}{Required-Ligatures}
 321 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rphf}{Reph-Forms}
 322 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtbd}{Right-Bounds}
 323 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtla}{Right-to-left-alternates}
 324 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtlm}{Right-to-left-mirrored-forms}
 325 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ruby}{Ruby-Notation-Forms}
 326 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rvrn}{Required-Variation-Alternates}
 327 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {salt}{Stylistic-Alternates}
 328 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sinf}{Scientific-Inferiors}
 329 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {size}{Optical-size}
 330 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smcp}{Small-Capitals}
 331 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smp1}{Simplified-Forms}
 332 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssNN}{Stylistic-Set-\$N\$}
 333 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssty}{Math-script-style-alternates}
 334 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {stch}{Stretching-Glyph-Decomposition}
 335 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {subs}{Subscript}
 336 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {supr}{Superscript}
 337 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {swsh}{Swash}
 338 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {titl}{Titling}
 339 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing-Jamo-Forms}
 340 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional-Name-Forms}
 341 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular-Figures}
 342 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {trad}{Traditional-Forms}
 343 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third-Widths}
 344 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {unic}{Unicase}
 345 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate-Vertical-Metrics}
 346 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu-Variants}

```

347 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical-Writing}
348 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate-Vertical-Half-Metrics}
349 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vjmo}{Vowel-Jamo-Forms}
350 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical-Kana-Alternates}
351 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkern}{Vertical-Kerning}
352 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional-Alternate-Vertical-Me}
353 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrt2}{Vertical-Alternates-and-Rotation}
354 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrtr}{Vertical-Alternates-for-Rotation}
355 \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 even whether certain features are available.

```

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

```

```

72         {
73         \int_case:nn \l_keys_choice_int { 5 {node} 6 {base} }
74         }
75 <debug>\typeout{Mode: \l_fontspec_mode_tl}
76         \tl_gset:Nx \g_@@_single_feat_tl { mode=\l_fontspec_mode_tl }
77 </LU>
78     }
79 }
80 }

```

1.2 Pre-parsed features

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

```

81 \@@_keys_define_code:nnn {fontspec-preparse} {Script}
82 {
83 <XE> \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
84 \tl_set:Nn \l_@@_script_name_tl {#1}
85 }

```

Exactly the same:

```

86 \@@_keys_define_code:nnn {fontspec-preparse} {Language}
87 {
88 <XE> \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
89 \tl_set:Nn \l_@@_lang_name_tl {#1}
90 }

```

TTC font index

```

91 \@@_keys_define_code:nnn {fontspec-preparse} {FontIndex}
92 {
93 \str_if_eq:eeF { \str_lower_case:f {\l_@@_extension_tl} } {.ttc}
94 { \@@_warning:n {font-index-needs-ttc} }
95 <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
96 <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
97 }
98 \@@_keys_define_code:nnn {fontspec} {FontIndex}
99 {
100 <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
101 <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
102 }

```

1.3 Font faces

Upright

```

103 \@@_keys_define_code:nnn {fontspec-preparse-external} {UprightFont}
104 {
105 \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
106 }

```

Italic and slanted

```

107 \@@_keys_define_code:nnn {fontspec-prepare-external} {ItalicFont}
108 {
109   \tl_if_empty:nTF {#1}
110   {
111     \bool_set_true:N \l_@@_noit_bool
112   }
113   {
114     \bool_set_false:N \l_@@_noit_bool
115     \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
116   }
117 }
118 \@@_keys_define_code:nnn {fontspec-prepare-external} {SlantedFont}
119 {
120   \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
121 }

```

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!

```

122 %\@@_keys_define_code:nnn {fontspec-prepare-external} {BoldSeries}
123 % {
124 %   \tl_gset:Nx \g_@@_curr_series_tl { #1 }
125 %   \seq_put_right:Nx \l_@@_bf_series_seq { #1 }
126 % }

```

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

```

127 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldFont}
128 {
129   \tl_if_empty:nTF {#1}
130   {
131     \bool_set_true:N \l_@@_nobf_bool
132   }
133   {
134     \bool_set_false:N \l_@@_nobf_bool
135     \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
136
137     \seq_if_empty:NT \l_@@_bf_series_seq
138     {
139       \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
140       \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
141     }
142
143     \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
144     {
145       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
146     }
147
148     \prop_put:NxV \l_@@_nfss_prop {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
149
150     <debug>\typeout{Setting-bold-font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}

```

```

151
152     }
153 }

```

Bold italic/slanted

```

154 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldItalicFont}
155 {
156     \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
157 }
158 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSlantedFont}
159 {
160     \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
161 }

```

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

```

162 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
163 {
164     \tl_if_empty:nTF {#1}
165     {
166         \bool_set_true:N \l_@@_nosc_bool
167     }
168     {
169         \bool_set_false:N \l_@@_nosc_bool
170         \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
171     }
172 }

```

1.3.1 Prepared font features

```

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

```

```

194     \clist_set:Nn \l_@@_fontfeat_sl_clist {#1}
195   }
196   \@@_keys_define_code:nnn {fontspec-preparse} {BoldSlantedFeatures}
197   {
198     \clist_set:Nn \l_@@_fontfeat_bfsl_clist {#1}
199   }

```

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

```

200   \@@_keys_define_code:nnn {fontspec} {SmallCapsFeatures}
201   {
202     \bool_if:NF \l_@@_firsttime_bool
203     {
204       \clist_set:Nn \l_@@_fontfeat_sc_clist {#1}
205     }
206   }

```

Features varying by size

```

207   \@@_keys_define_code:nnn {fontspec-preparse} {SizeFeatures}
208   {
209     \clist_set:Nn \l_@@_sizefeat_clist {#1}
210     \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
211   }
212   \@@_keys_define_code:nnn {fontspec-preparse-nested} {SizeFeatures}
213   {
214     \clist_set:Nn \l_@@_sizefeat_clist {#1}
215     \tl_if_empty:NT \l_@@_this_font_tl
216     { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
217   }
218   \@@_keys_define_code:nnn {fontspec-preparse-nested} {Font}
219   {
220     \tl_set:Nn \l_@@_this_font_tl {#1}
221   }
222   \@@_keys_define_code:nnn {fontspec} {SizeFeatures}
223   {
224     % dummy
225   }
226   \@@_keys_define_code:nnn {fontspec} {Font}
227   {
228     % dummy
229   }
230   \@@_keys_define_code:nnn {fontspec-sizing} {Size}
231   {
232     \tl_set:Nn \l_@@_size_tl {#1}
233   }
234   \@@_keys_define_code:nnn {fontspec-sizing} {Font}
235   {
236     \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
237   }

```

1.4 General font-independent features

These features can be applied to any font.

NFSS encoding For the very brave.

```

238 \@@_keys_define_code:nnn {fontspec-preparse} {NFSSEncoding}
239 {
240   \tl_gset:Nx \g_@@_nfss_enc_tl { #1 }
241 }

```

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.)

```

242 \@@_keys_define_code:nnn {fontspec-preparse} {NFSSFamily}
243 {
244   \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
245 }

```

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

```

246 \@@_keys_define_code:nnn {fontspec} {FontFace}
247 {
248   \tl_clear:N \l_@@_this_font_tl
249   \clist_set:No \l_@@_arg_clist { \use_iii:nnn #1 }
250   \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
251   \int_compare:nT { \clist_count:N \l_@@_arg_clist = 1 }
252   {
253     (debug)\typeout{FontFace~ parsing:~ one~ clist~ item}
254     \tl_if_in:NnF \l_@@_arg_clist {=}
255     {
256       (debug)\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
257       \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
258       \tl_clear:N \l_@@_this_feat_clist
259     }
260   }
261
262   \@@_add_nfssfont:nnnn
263   { \use_i:nnn #1 } { \use_ii:nnn #1 } { \l_@@_this_font_tl } { \l_@@_this_feat_clist }
264 }

```

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.

```

265 \@@_keys_define_code:nnn {fontspec} {Scale}
266 {
267   \str_case:nnF {#1}
268   {
269     {MatchLowercase} { \@@_calc_scale:n {5} }
270     {MatchUppercase} { \@@_calc_scale:n {8} }
271   }
272   { \tl_set:Nx \l_@@_scale_tl {#1} }
273 }

```

ScaleAgain

```

274 \@@_keys_define_code:nnn {fontspec} {ScaleAgain}
275 {
276   \tl_if_empty:NT \l_@@_scale_tl { \tl_set:Nn \l_@@_scale_tl {1} }
277   \tl_set:Nx \l_@@_scale_tl { \fp_eval:n { #1 * \l_@@_scale_tl } }
278   \@@_info:n {set-scale}
279 }

```

`\@@_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.)

```

280 \cs_new:Nn \@@_calc_scale:n
281 {
282   \group_begin:
283
284   \fontencoding { \encodingdefault }
285   \fontfamily { \rmdefault }
286   \selectfont
287
288   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
289   \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_@@_fontface_cs_tl
290
291   \tl_set:Nx \l_@@_scale_tl
292   {
293     \fp_eval:n { \dim_to_fp:n { \l_@@_tmpa_dim } /
294                 \dim_to_fp:n { \l_@@_tmpb_dim } }
295   }
296
297   \@@_info:n {set-scale}
298   \exp_args:NNNx
299   \group_end:
300   \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
301 }

```

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

`\@@_set_font_dimen:NnN` This function sets the dimension #1 (for font #3) to ‘fontdimen’ #2 for either font dimension 5 (x-height) or 8 (cap-height). If, for some reason, these return an incorrect ‘zero’ value (as `\fontdimen8` might for a .t_{fm} 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’.

```

302 \cs_new:Nn \@@_set_font_dimen:NnN
303 {
304   \dim_set:Nn #1 { \fontdimen #2 #3 }

```



```

305 \dim_compare:nNnT #1 = {0pt}
306 {
307   \settoheight #1
308   {
309     \str_if_eq:nnTF {#3} {\font} \rmfamily #3
310     \int_case:nnF #2
311     {
312       {5} {x} % x-height
313       {8} {X} % cap-height
314       } {?} % "else" clause; never reached.
315     }
316   }
317 }

```

(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.

```

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

```

324 \cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
325 {
326   \tl_if_empty:nTF {#4}
327   {
328     \tl_set:Nn \l_@@_wordspace_adjust_tl
329     {
330       \fontdimen 2 \font = #1 \fontdimen 2 \font
331       \fontdimen 3 \font = #1 \fontdimen 3 \font
332       \fontdimen 4 \font = #1 \fontdimen 4 \font
333     }
334   }
335   {
336     \tl_set:Nn \l_@@_wordspace_adjust_tl
337     {
338       \fontdimen 2 \font = #1 \fontdimen 2 \font
339       \fontdimen 3 \font = #2 \fontdimen 3 \font
340       \fontdimen 4 \font = #3 \fontdimen 4 \font
341     }
342   }
343 }

```

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

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

```

344 \@@_keys_define_code:nnn {fontspec} {PunctuationSpace}
345 {
346   \str_case_e:nnF {#1}
347   {
348     {WordSpace}
349     {
350       \tl_set:Nn \l_@@_punctspace_adjust_tl
351       { \fontdimen 7 \font = 0 \fontdimen 2 \font }
352     }
353     {TwiceWordSpace}
354     {
355       \tl_set:Nn \l_@@_punctspace_adjust_tl
356       { \fontdimen 7 \font = 1 \fontdimen 2 \font }
357     }
358   }
359   {
360     \tl_set:Nn \l_@@_punctspace_adjust_tl
361     { \fontdimen 7 \font = #1 \fontdimen 7 \font }
362   }
363 }
364 \@@_aff_error:n {PunctuationSpace}

```

Secret hook into the font-adjustment code

```

365 \@@_keys_define_code:nnn {fontspec} {FontAdjustment}
366 {
367   \tl_put_right:Nx \l_@@_postadjust_tl {#1}
368 }

```

Letterspacing

```

369 \@@_keys_define_code:nnn {fontspec} {LetterSpace}
370 {
371   \@@_update_featstr:n {letterspace=#1}
372 }

```

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.

```

373 \@@_keys_define_code:nnn {fontspec} {HyphenChar}
374 {
375   \str_if_eq:nnTF {#1} {None}
376   {
377     \tl_put_right:Nn \l_@@_postadjust_tl
378     { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
379   }
380   {
381     \@@_warning:nx {only-xetex-feature} {HyphenChar}
382   }

```

```

383 \tl_if_single:nTF {#1}
384 { \tl_set:Nn \l_@@_hyphenchar_tl {\`#1} }
385 { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }
386
387 \exp_args:No \@@_primitive_font_glyph_if_exist:NnTF \l_@@_fontface_cs_tl {\l_@@_hyphen
388 {
389 \tl_put_right:Nn \l_@@_postadjust_tl
390 { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
391 }
392 { \@@_error:nx {no-glyph}{\l_fontspeg_fontname_tl}{#1} }
393
394 }
395 }
396 \@@_aff_error:n {HyphenChar}

```

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

```

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

```

```

431 \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
432 {
433   \bool_if:NF \l_@@_firsttime_bool
434   { \@@_warning:nx {opa-twice} {#1} }
435 }
436 \tl_set:Nx \l_@@_opacity_tl
437 {
438   \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
439   \int_to_hex:n { \l_@@_tmp_int }
440 }
441 }

```

Mapping

```

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

```

1.4.1 Continuous font axes

```

464 \@@_keys_define_code:nnn {fontspec} {Weight}
465 {
466   \@@_update_featstr:n{weight=#1}
467 }
468 \@@_keys_define_code:nnn {fontspec} {Width}
469 {
470   \@@_update_featstr:n{width=#1}
471 }
472 \@@_keys_define_code:nnn {fontspec} {OpticalSize}
473 < *XE>
474 {
475   \bool_if:NTF \l_@@_ot_bool
476   {

```

```

477     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
478   }
479   {
480     \bool_if:NT \l_@@_mm_bool
481     {
482       \@@_update_featstr:n { optical size = #1 }
483     }
484   }
485   \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
486   {
487     \bool_if:NT \l_@@_firsttime_bool
488     { \@@_warning:n {no-opticals} {\l_fontspeg_fontname_tl} }
489   }
490 }
491 </XE>
492 <*LU>
493 {
494   \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
495 }
496 </LU>

```

1.4.2 Font transformations

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

```

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

```

```

525 {
526   FakeBold .code:n = { \l_@@_warning:n {fakebold-only-xetex} }
527 }
528 \end{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.)

```

529 \keys_define:nn {fontspec}
530 {
531   AutoFakeSlant .code:n =
532   {
533     \bool_if:NT \l_@@_firsttime_bool
534     {
535       \tl_set:Nn \l_@@_fake_slant_tl {#1}
536       \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
537       \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
538       \bool_set_false:N \l_@@_noit_bool
539
540       \tl_if_empty:NF \l_@@_fake_embolden_tl
541       {
542         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
543         {FakeBold=\l_@@_fake_embolden_tl}
544         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
545         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
546       }
547     }
548   },
549   AutoFakeSlant .default:n = {\Q.2}
550 }

```

Same but reversed:

```

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

```

```

567             \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
568         }
569     }
570 },
571 AutoFakeBold .default:n = {1.5}
572 }

```

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.

```

573 \@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}
574 {
575     \@@_update_featstr:n {#1}
576 }
577 \@@_keys_define_code:nnn {fontspec-aat} {RawFeature}
578 {
579     \@@_update_featstr:n {#1}
580 }

```

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  \langle XE \rangle mapping = tex-text
11  \langle LU \rangle +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 \langle *XE \rangle
20 \keys_define:nn {fontspec-opentype}
21 {
22   Ligatures / TeX .code:n = { \tl_set:Nn \l_@@_mapping_tl {tex-text} },
23   Ligatures / TeX .groups:n = {opentype},
24   Ligatures / TeXReset .code:n = { \tl_clear:N \l_@@_mapping_tl },
25   Ligatures / TeXReset .groups:n = {opentype},
26 }
27 \langle /XE \rangle
28 \langle LU \rangle \@@_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   +sups,-sups,
65   +subs,-subs,
66   +ordn,-ordn,
67   +numr,-numr,
68   +dnom,-dnom,
69   +sinf,-sinf,
70 }
71 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Superior} {sups} {sups} {+s}
72 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Inferior} {subs} {subs} {+s}
73 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Ordinal} {ordn} {ordn} {+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} {On} {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}

```

2.9 Style

```

125 \@@_define_opentype_feature_group:n {Style}
126 \@@_define_opentype_feature:nnnnn {Style} {ResetAll} {} {}
127 {
128     +salt,-salt,
129     +ital,-ital,
130     +ruby,-ruby,
131     +swsh,-swsh,
132     +hist,-hist,
133     +titl,-titl,
134     +hkna,-hkna,
135     +vkna,-vkna,
136     +ssty=0,-ssty=0,
137     +ssty=1,-ssty=1,
138 }

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

```

2.10 CJK shape

```

151 \@@_define_opentype_feature_group:n {CJKShape}
152 \@@_define_opentype_feature:nnnnn {CJKShape} {ResetAll} {} {}
153 {
154     +trad,-trad,
155     +smpl,-smpl,
156     +jp78,-jp78,
157     +jp83,-jp83,
158     +jp90,-jp90,
159     +jp04,-jp04,
160     +expt,-expt,
161     +nlck,-nlck,
162 }

163 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Traditional} {trad} {trad} {+smpl,+jp78,+jp83}
164 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Simplified} {smpl} {smpl} {+trad,+jp78,+jp83}

```

```

165 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978} {jp78} {jp78} {+trad,+smpl,+jp83}
166 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983} {jp83} {jp83} {+trad,+smpl,+jp78}
167 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990} {jp90} {jp90} {+trad,+smpl,+jp78}
168 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS2004} {jp04} {jp04} {+trad,+smpl,+jp78}
169 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Expert} {expt} {expt} {+trad,+smpl,+jp78}
170 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {NLC} {nlck} {nlck} {+trad,+smpl,+jp78}

```

2.11 Character width

```

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

```

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

```

3 OpenType features that need numbering

3.1 Alternate

```

198 \@@_define_opentype_feature_group:n {Alternate}
199 \keys_define:nn {fontspec-opentype}
200 {
201     Alternate .default:n = {} ,
202     Alternate .groups:n = {opentype},

```

```

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

```

3.2 Variant / StylisticSet

```

217 \@@_define_opentype_feature_group:n {Variant}
218 \keys_define:nn {fontspec-opentype}
219 {
220     Variant .default:n = {0} ,
221     Variant .groups:n = {opentype} ,
222     Variant / unknown .code:n =
223     {
224         \clist_map_inline:nn {#1}
225         {
226             \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
227         }
228     }
229 }
230 \aliasfontfeature{Variant}{StylisticSet}

```

3.3 CharacterVariant

```

231 \@@_define_opentype_feature_group:n {CharacterVariant}
232 \use:x
233 {
234     \cs_new:Npn \exp_not:N \fontspec_parse_cv:w
235     ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
236     {
237         \@@_make_OT_feature:xxx
238         { cv \exp_not:N \two@digits {##1} }
239         { +cv \exp_not:N \two@digits {##1} = ##2 } {}
240     }
241     \keys_define:nn {fontspec-opentype}
242     {
243         CharacterVariant / unknown .code:n =
244         {
245             \clist_map_inline:nn {##1}
246             {
247                 \exp_not:N \fontspec_parse_cv:w
248                 #####1 \c_colon_str 0 \c_colon_str \exp_not:N \q_nil

```

```

249         }
250     }
251 }
252 }

```

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

3.4 Annotation

```

253 \@@_define_opentype_feature_group:n {Annotation}
254 \keys_define:nn {fontspec-opentype}
255 {
256     Annotation .default:n = {0} ,
257     Annotation .groups:n = {opentype},
258     Annotation / unknown .code:n =
259     {
260         \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
261     }
262 }

```

3.5 Ornament

```

263 \@@_define_opentype_feature_group:n {Ornament}
264 \keys_define:nn {fontspec-opentype}
265 {
266     Ornament .default:n = {0} ,
267     Ornament .groups:n = {opentype},
268     Ornament / unknown .code:n =
269     {
270         \@@_make_OT_feature:nnn {ornm} {+ornm=#1} {}
271     }
272 }

```

4 Script and Language

4.1 Script

```

273 \keys_define:nn {fontspec-opentype}
274 {
275     Script .choice: ,
276     Script .groups:n = {opentype} ,
277 }
278 \cs_new:Nn \fontspec_new_script:nn
279 {
280     \keys_define:nn {fontspec-opentype} { Script / #1 .code:n =
281     {
282         <debug>\typeout{Trying~[Script=#1]}
283         \bool_set_false:N \l_@@_scriptlang_exist_bool
284         \clist_map_inline:nn {#2}
285         {
286             \exp_args:No \@@_check_script:NnT \l_@@_fontface_cs_tl {####1}
287             {
288                 <debug>\typeout{Script~tag~found:~####1}
289                 \tl_set:Nn \l_@@_script_name_tl {#1}

```

```

290         \tl_set:Nn \l_fontspec_script_tl {###1}
291         \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
292         \bool_set_true:N \l_@@_scriptlang_exist_bool
293         \tl_gset:Nx \g_@@_single_feat_tl { script=###1 }
294         \clist_map_break:
295     }
296 }
297 \bool_if:NF \l_@@_scriptlang_exist_bool
298 {
299     \bool_if:nF { \str_if_eq_p:ee {#1} {CustomDefault} }
300     {
301         \tl_clear:N \l_@@_script_name_tl
302         \@@_warning:nx {no-script} {\l_fontspec_fontname_tl} {#1}
303     }
304
305     \bool_if:nF
306     {
307         \str_if_eq_p:ee {#1} {Default} ||
308         \str_if_eq_p:ee {#1} {Latin} ||
309         \str_if_eq_p:ee {#1} {CustomDefault}
310     }
311     {
312         \keys_set:nn {fontspec-opentype} { Script = CustomDefault }
313     }
314 }
315 }
316 }
317 }

```

4.2 Language

```

318 \keys_define:nn {fontspec-opentype}
319 {
320     Language .choice: ,
321     Language .groups:n = {opentype} ,
322 }
323 \cs_new:Nn \fontspec_new_lang:nn
324 {
325     \keys_define:nn {fontspec-opentype} { Language / #1 .code:n =
326     {
327         \bool_set_false:N \l_@@_scriptlang_exist_bool
328         \clist_map_inline:nn {#2}
329         {
330             \exp_args:No \@@_check_lang:NnTF \l_@@_fontface_cs_tl {#2}
331             {
332                 \tl_set:Nn \l_fontspec_lang_tl {#2}
333                 \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
334                 \tl_gset:Nx \g_@@_single_feat_tl { language=#2 }
335                 \bool_set_true:N \l_@@_scriptlang_exist_bool
336                 \clist_map_break:
337             }
338         }
339     }

```

```

339     \bool_if:NF \l_@@_scriptlang_exist_bool
340     {
341         \@@_warning:nx {language-not-exist} {#1}
342         \keys_set:nn {fontspec-opentype} { Language = Default }
343     }
344 }
345 }
346 }

```

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.

```

347 \@@_keys_define_code:nnn {fontspec-opentype} { Language / Default }
348 {
349     \tl_set:Nn \l_fontspec_lang_tl {DFLT}
350     \int_zero:N \l_@@_language_int
351     \tl_gset:Nn \g_@@_single_feat_tl { language=DFLT }
352 }

```

5 Backwards compatibility

```

353 \cs_new:Nn \@@_ot_compat:nn
354 {
355     \aliasfontfeatureoption {#1} {#2off} {No#2}
356 }
357 \@@_ot_compat:nn {Ligatures} {Rare}
358 \@@_ot_compat:nn {Ligatures} {Required}
359 \@@_ot_compat:nn {Ligatures} {Common}
360 \@@_ot_compat:nn {Ligatures} {Discretionary}
361 \@@_ot_compat:nn {Ligatures} {Contextual}
362 \@@_ot_compat:nn {Ligatures} {Historic}
363 \@@_ot_compat:nn {Numbers} {SlashedZero}
364 \@@_ot_compat:nn {Contextuals} {Swash}
365 \@@_ot_compat:nn {Contextuals} {Alternate}
366 \@@_ot_compat:nn {Contextuals} {WordInitial}
367 \@@_ot_compat:nn {Contextuals} {WordFinal}
368 \@@_ot_compat:nn {Contextuals} {LineFinal}
369 \@@_ot_compat:nn {Contextuals} {Inner}
370 \@@_ot_compat:nn {Diacritics} {MarkToBase}
371 \@@_ot_compat:nn {Diacritics} {MarkToMark}
372 \@@_ot_compat:nn {Diacritics} {AboveBase}
373 \@@_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{Manichaeian}{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{Provençal}{PRO}
 285 \newfontlanguage{Portuguese}{PTG}
 286 \newfontlanguage{Chin}{QIN}
 287 \newfontlanguage{Rajasthani}{RAJ}
 288 \newfontlanguage{R-Cree}{RCR}
 289 \newfontlanguage{Russian~Buriat}{RBU}
 290 \newfontlanguage{Riang}{RIA}
 291 \newfontlanguage{Rhaeto-Romanic}{RMS}
 292 \newfontlanguage{Romanian}{ROM}
 293 \newfontlanguage{Romany}{ROY}
 294 \newfontlanguage{Rusyn}{RSY}
 295 \newfontlanguage{Ruanda}{RUA}
 296 \newfontlanguage{Russian}{RUS}
 297 \newfontlanguage{Sadri}{SAD}
 298 \newfontlanguage{Sanskrit}{SAN}
 299 \newfontlanguage{Santali}{SAT}

```

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

1.1 Ligatures

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

Emulate CM extra ligatures.

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

1.2 Letters

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

1.3 Numbers

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

Both naming conventions are offered to select the number case.

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

1.4 Contextuals

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

1.5 Diacritics

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

1.6 Vertical position

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

1.7 Fractions

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

1.8 Alternate

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

```

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

```

1.9 Variant / StylisticSet

```

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

```

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@antt@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   \bool_if:NT \g_@@_math_bool
127   {
128     \@@_info:n {setup-math}
129     \fontspec_setup_maths:
130   }

```

```
131 }  
132 \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_AT_EX's 'shape' font axis needs to be overloaded to support italic small caps and slanted small caps. These are the combinations to support:

```
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 `\-`

`\-` This macro is courtesy of Frank Mittelbach and the \LaTeX 2_ε source code.

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

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

1.2 Verbatims

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`verbatim*` With the `verbatim` package.

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

This is for vanilla \LaTeX .

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

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

```
66 \cs_set:Npn \fontspec_patch_moreverb:
67 {
68   \@ifpackageloaded{moreverb}
69   {
70     \cs_set:cpn {listingcont*}
71     {
72       \cs_set:Npn \verbatim@processline
73       {
74         \thelisting@line \global\advance\listing@line1\relax
75         \the\verbatim@line\par
76       }
77       \@verbatim \fontspec_print_visible_spaces: \verbatim@start
78     }
79   }{}
80 }
```

listings and fancvrb make things nice and easy:

```
81 \cs_set:Npn \fontspec_patch_fancyvrb:
82 {
83   \@ifpackageloaded{fancyvrb}
84   {
85     \cs_set_eq:NN \FancyVerbSpace \fontspec_visible_space:
86   }{}
87 }
88 \cs_set:Npn \fontspec_patch_listings:
89 {
90   \@ifpackageloaded{listings}
91   {
92     \cs_set_eq:NN \lst@visibleSPACE \fontspec_visible_space:
93   }{}
94 }
95 </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	
\#	262, 263, 294
\,	1, 2, 3, 4, 5, 17
\-	2, 4, 126, 648
@@ commands:	
\@@_DeclareFontShape:nnnnnn	498, 505, 515, 533
\g_@@_OT_features_prop	9, 11, 71
\@@_add_nfssfont:nnnn	262, 305, 306, 307, 308, 309, 310, 325
\@@_aff_error:n	11, 323, 364, 396
\l_@@_alias_bool	21, 204, 211, 217, 222, 229, 249
\l_@@_all_features_clist	21, 53, 99, 109, 123, 181, 271
\g_@@_all_keyval_modules_clist	1, 47, 206, 224
\g_@@_all_opentype_feature_names_prop	72, 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
\l_@@_arg_clist	59, 249, 250, 251, 254, 257
\l_@@_atsui_bool	7, 10, 204, 344, 353, 606
\l_@@_basename_tl	10, 42, 90, 322
\l_@@_bf_series_seq	45, 125, 137, 140
\g_@@_bfmathrm_tl	69, 70, 82, 87, 88, 89, 122
\@@_calc_scale:n	269, 270, 280
\g_@@_cfg_bool	1, 7, 8, 16
\l_@@_check_bool	5, 65, 66, 204, 209, 215, 229
\l_@@_check_feat_bool	27, 57, 59, 60
\@@_check_lang:Nn	127
\@@_check_lang:NnTF	97, 127, 330
\@@_check_lang:Nnn	131
\@@_check_lang:NnnTF	110, 127
\@@_check_ot_feat:Nn	178
\@@_check_ot_feat:NnTF	50, 62, 178, 598
\@@_check_ot_feat:Nnnn	183
\@@_check_ot_feat:NnnnTF	67, 178
\@@_check_script:Nn	81
\@@_check_script:NnTF	80, 81, 286
\@@_combo_sc_shape:n	506, 509, 554, 562
\@@_construct_font_call:nn	136, 138, 142, 145, 151, 171, 274, 391, 392, 413, 492
\@@_construct_font_call:nnnnnn	151, 160
\l_@@_curr_bfname_tl	92, 135, 145, 148, 150, 182
\l_@@_curr_fontname_tl	91, 316, 317
\g_@@_curr_series_tl	85, 124, 139, 143, 148, 150, 182, 644
\@@_declare_shape:nnnn	402, 421
\@@_declare_shape_loginfo:nn	433, 537
\@@_declare_shape_slanted:nn	432, 525
\@@_declare_shapes_normal:nn	430, 496
\@@_declare_shapes_smcaps:nn	431, 501
\g_@@_default_fontopts_clist	46, 111, 119
\@@_define_aat_feature:nnnn	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 60, 61, 62, 63, 65, 66, 67, 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
 . . 5, 7, 30, 43, 43, 44, 45, 49, 50, 62,
 78, 94, 106, 112, 113, 114, 116, 121,
 122, 123, 126, 149, 150, 152, 172, 193
 \@@_define_opentype_feature_
 group:n 1, 6,
 29, 42, 61, 77, 93, 105, 115, 125, 151,
 171, 188, 189, 198, 217, 231, 253, 263
 \@@_define_opentype_onoffreset:nnnnn
 13, 14, 15, 16, 17, 18,
 35, 36, 37, 38, 39, 39, 40, 41, 52, 53,
 54, 55, 56, 60, 71, 72, 73, 74, 75, 76,
 87, 88, 89, 90, 91, 92, 101, 102, 103,
 104, 111, 124, 139, 140, 141, 142,
 143, 144, 145, 146, 147, 148, 163,
 164, 165, 166, 167, 168, 169, 170,
 182, 183, 184, 185, 186, 187, 188,
 190, 191, 192, 193, 194, 195, 196, 197
 \@@_define_opentype_onreset:nnnnn
 28, 47
 \g_@@_defined_shapes_tl
 86, 184, 539, 643
 \l_@@_defining_encoding_bool . 11,
 17, 23, 23, 29, 35, 59, 61, 66, 72, 78, 84
 \l_@@_disable_defaults_bool 20, 97, 153
 \l_@@_em_int
 . 36, 93, 96, 97, 100, 102, 105, 156, 159
 \g_@@_em_normalise_slant_bool . .
 25, 62, 68, 88
 \g_@@_em_prop . . . 60, 73, 77, 78, 94, 102
 \l_@@_em_switch_tl 102, 103, 115
 \l_@@_em_tmp_tl 94, 96, 116
 \l_@@_emdef_int 37, 61, 77, 78, 79
 \l_@@_emshape_query_tl
 86, 90, 94, 97, 114
 \@@_error:n 1, 10, 443
 \@@_error:nn . 2, 3, 12, 14, 18, 24, 30,
 36, 67, 73, 79, 85, 140, 392, 414, 679, 698
 \g_@@_euenc_bool 9, 10, 18, 23, 36, 39, 58
 \l_@@_ext_filename_tl 85, 86, 89, 90, 93
 \l_@@_extension_tl
 39, 45, 53, 72, 93, 94, 162
 \l_@@_extensions_clist 48, 50, 61, 254
 \l_@@_external_bool . . . 22, 28, 40, 375
 \@@_extract_all_features: 94
 \@@_extract_all_features:n . . . 20, 94
 \l_@@_fake_embolden_tl
 130, 540, 543, 557
 \l_@@_fake_slant_tl . 129, 535, 562, 565
 \l_@@_family_fontopts_clist
 52, 105, 106, 112
 \g_@@_family_int_prop . . . 76, 248, 254
 \l_@@_family_label_tl
 105, 107, 128, 149, 161
 \@@_feat_off:n 39, 44
 \@@_feat_prop_add:nn . 1, 2, 3, 4, 5, 5, 17
 \@@_feat_reset:n 40, 45, 50
 \@@_find_autofonts: 262, 282
 \l_@@_firsttime_bool
 1, 29, 179, 202, 271, 320,
 410, 421, 433, 487, 533, 555, 617, 637
 \@@_font_is_file: 30, 168, 170
 \@@_font_is_name: 168, 638
 \l_@@_font_path_tl . . . 29, 95, 174, 639
 \@@_font_suppress_not_found_
 error: 5, 9, 38, 267
 \l_@@_fontcfg_bool . . . 11, 12, 18, 22, 81
 \l_@@_fontface_cs_tl 17,
 62, 144, 148, 148, 153, 156, 157, 160,
 164, 286, 289, 330, 387, 412, 598, 609
 \l_@@_fontfeat_bf_clist
 62, 179, 306, 558
 \l_@@_fontfeat_bfit_clist
 64, 190, 309, 542, 544, 564, 566
 \l_@@_fontfeat_bfsl_clist 66, 198, 310
 \l_@@_fontfeat_clist 57, 130, 193, 272
 \l_@@_fontfeat_curr_clist
 58, 452, 461, 474
 \l_@@_fontfeat_it_clist
 63, 186, 307, 536
 \l_@@_fontfeat_sc_clist . 67, 204, 452
 \l_@@_fontfeat_sl_clist . 65, 194, 308
 \l_@@_fontfeat_up_clist
 61, 175, 210, 305
 \g_@@_fontid_family_prop 75, 228, 256
 \l_@@_fontid_tl
 21, 23, 42, 96, 224, 228, 236
 \l_@@_fontname_bf_tl
 75, 132, 145, 287, 293, 306, 559
 \l_@@_fontname_bfit_tl 76,
 134, 156, 286, 287, 288, 309, 545, 567
 \l_@@_fontname_bfsl_tl
 136, 160, 301, 310
 \l_@@_fontname_it_tl
 74, 115, 133, 286, 298, 307, 537
 \l_@@_fontname_sc_tl 137, 170, 456, 468
 \l_@@_fontname_sl_tl 120, 135, 301, 308
 \l_@@_fontname_tl . . . 97, 152, 154, 158
 \l_@@_fontname_up_tl 9, 42,
 45, 105, 126, 131, 136, 138, 140, 142, 145

\@@_fontname_wrap:n	
.....	47, 153, 154, 170, 174
\l_@@_fontopts_clist	
..	51, 102, 103, 113, 402, 409, 410, 411
\g_@@_fontopts_prop	
.....	68, 87, 102, 105, 132, 135, 139, 140, 409
\@@_get_features:Nn	59, 189
\@@_get_features:n ..	28, 189, 273, 481
\l_@@_graphite_bool ..	10, 204, 347, 365
\c_@@_hexcol_tl	148, 217, 659
\l_@@_hexcol_tl	
.....	98, 216, 218, 401, 405, 418, 659
\l_@@_hyphenchar_tl	
.....	119, 384, 385, 387, 390
\@@_if_autofont:nn	388
\@@_if_autofont:nnTF	381
\@@_if_detect_external:n	58
\@@_if_detect_external:nTF ..	12, 58, 170
\@@_if_font_feature:n	263
\@@_if_font_feature:nTF	261
\@@_if_merge_shape:n	25
\@@_if_merge_shape:nTF	21, 184
\@@_info:n	7, 128, 278, 297
\@@_info:nn	8, 383
\@@_info:nnn	9, 265
\@@_init:	6, 169, 268, 633
\@@_init_fontface:	192, 654
\@@_init_ttc:n	17, 70
\@@_int_mult_truncate:Nn	74, 430
\@@_iv_str_to_num:Nn	
.....	92, 141, 142, 193, 194, 196, 682
\@@_iv_str_to_num:w	692, 693, 695
\@@_keys_define_code:nnn	7,
	13, 16, 20, 24, 36, 37, 46, 81, 86, 91,
	98, 103, 107, 118, 122, 127, 154, 158,
	162, 173, 177, 184, 188, 192, 196,
	200, 207, 212, 218, 222, 226, 230,
	234, 238, 242, 246, 265, 274, 318,
	344, 347, 365, 369, 373, 397, 427,
	443, 447, 453, 464, 468, 472, 573, 577
\l_@@_keys_leftover_clist	
.....	55, 124, 127, 128, 129,
	194, 195, 199, 200, 203, 205, 209, 210
\@@_keys_set_known:nnN	
..	67, 122, 127, 129, 193, 195, 331, 400
\@@_lang_dflt_correct:N	195, 709
\l_@@_lang_name_tl	
.....	89, 131, 141, 142, 183, 185
\l_@@_language_int	
.....	30, 45, 194, 195, 201, 207, 278, 333, 350
\l_@@_leftover_clist	54, 400, 402
\@@_load_external_fontoptions:Nn	
.....	18, 79, 408
\@@_load_font:	26, 132
\@@_load_fontname:Nn ..	401, 405, 447, 468
\@@_main_DeclareFontExtensions:n	
.....	106, 252
\@@_main_IfFontFeatureActiveTF:nnn	
.....	110, 257
\@@_main_addfontfeatures:n ..	66, 70, 144
\@@_main_aliasfontfeature:nn ..	90, 201
\@@_main_aliasfontfeatureoption:nnn	
.....	94, 220
\@@_main_fontspec:nn	1, 3
\@@_main_liningnums:n	121, 292
\@@_main_newAATfeature:nnnn ..	78, 175
\@@_main_newfontface:nnn	55, 112
\@@_main_newfontfamily:nnnN	
.....	43, 47, 51, 99
\@@_main_newfontfeature:nn	74, 168
\@@_main_newopentypefeature:nnn	
.....	82, 86, 185
\@@_main_oldstylenums:n	116, 285
\@@_main_setboldmathrm:nn	27, 67
\@@_main_setmainfont:nn	7, 8, 39
\@@_main_setmathrm:nn	23, 61
\@@_main_setmathsf:nn	31, 73
\@@_main_setmathtt:nn	35, 79
\@@_main_setmonofont:nn	18, 43
\@@_main_setsansfont:nn	13, 25
\@@_make_AAT_feature:nn ..	9, 12, 76, 87
\@@_make_AAT_feature_string:Nnn ..	26
\@@_make_AAT_feature_string:NnnTF	
.....	12, 17, 26, 608
\@@_make_OT_feature:nnn	
.....	33, 52, 77, 206, 213, 226, 237, 260, 270
\@@_make_font_shapes:Nnnnn ..	317, 397
\@@_make_ot_smallcaps:TF	595
\@@_make_smallcaps:TF ..	458, 595, 601
\l_@@_mapping_tl	
.....	22, 24, 26, 144, 213, 214, 445, 449
\g_@@_math_bool	5, 6,
	17, 98, 100, 101, 102, 103, 104, 105,
	106, 107, 110, 112, 113, 116, 118,
	119, 120, 121, 122, 123, 124, 125, 126
\g_@@_math_euler_bool	9, 13, 34
\g_@@_math_lucida_bool	
.....	13, 14, 14, 15, 40
\g_@@_mathrm_tl	63,
.....	64, 74, 75, 77, 78, 81, 84, 96, 121, 125

\g_@@_mathsf_tl	\@@_preparse_features:
..... 75, 76, 79, 91, 97, 123, 126	25, 118
\g_@@_mathtt_tl	\l_@@_prev_unicode_name_tl ...
..... 80, 81, 82, 92, 98, 124, 127	57, 62
\l_@@_mm_bool 9, 346, 357, 480, 485	\l_@@_primitive_font
\@@_msg_new:nnn	39, 40
. 13, 18, 23, 44, 84, 90, 94, 104, 108,	\@@_primitive_font_current_name:
112, 117, 122, 127, 133, 137, 143, 56, 179
147, 151, 155, 160, 164, 169, 174,	\@@_primitive_font_get_name:N
178, 186, 190, 194, 198, 202, 207, 212	56, 393
\@@_msg_new:nnnn 15, 28, 37, 48, 58, 66, 74	\@@_primitive_font_glyph_if_-
\l_@@_never_check_bool	exist:Nn
..... 28, 84, 134, 185, 270	44
\g_@@_nfss_enc_tl 4, 15, 21, 33, 39, 51,	\@@_primitive_font_glyph_if_-
57, 87, 107, 240, 263, 498, 505, 533, 645	exist:NnTF
\l_@@_nfss_fam_tl .. 101, 226, 241, 244	16, 44, 387
\g_@@_nfss_family_tl	\@@_primitive_font_gset:Nnn
... 41, 88, 161, 180, 230, 241, 242, 1, 26, 28, 33
255, 256, 263, 269, 270, 271, 272,	\@@_primitive_font_gset:NnnTF ...
277, 278, 279, 280, 498, 505, 533, 534	34
\l_@@_nfss_prop	\@@_primitive_font_gset:NnnTTF .
69, 148, 181	21
\l_@@_nfss_sc_tl	\@@_primitive_font_gset:Onn ..
..... 99, 425, 473, 503, 506, 564	33, 144
\l_@@_nfss_tl .. 100, 424, 448, 499, 552	\@@_primitive_font_gset:OnnTF ...
\l_@@_nfssfont_prop	34
70, 312, 335	\@@_primitive_font_gset:p:NnnTF .
\l_@@_nobf_bool	21
..... 2, 26, 131, 134, 284, 291, 560	\@@_primitive_font_if_exist:nTF
\l_@@_noit_bool 35, 171
..... 3, 27, 111, 114, 284, 296, 538	\@@_primitive_font_if_null:NnTF .
\l_@@_nosc_bool 4, 166, 169, 454, 465, 471 13, 24, 29, 40
\c_@@_opacity_tl 149, 217, 419, 431, 658	\@@_primitive_font_if_null_p:N ..
\l_@@_opacity_tl	13
. 102, 216, 218, 419, 424, 431, 436, 658	\@@_primitive_font_set:Nnn
\l_@@_optical_size_tl 1, 21, 23, 31, 39, 391, 392
..... 103, 165, 477, 494, 640	\@@_primitive_font_set:NnnTF .
\l_@@_options_tl ... 104, 151, 154, 158	32, 137
\l_@@_ot_bool	\@@_primitive_font_set:NnnTTF ..
8, 28, 39,	21
65, 78, 91, 108, 121, 136, 197, 269,	\@@_primitive_font_set:Onn
345, 361, 370, 475, 485, 575, 603, 636	31
\@@_ot_compat:nn ... 353, 357, 358,	\@@_primitive_font_set:OnnTF .
359, 360, 361, 362, 363, 364, 365,	32, 412
366, 367, 368, 369, 370, 371, 372, 373	\@@_primitive_font_set_hyphenchar:Nn
\@@_ot_validate_tag:n 52, 378, 390
..... 108, 164, 165, 219, 220, 221, 664	\@@_primitive_font_set_p:NnnTF ..
\@@_ot_validate_tag:w	21
667, 670	\l_@@_proceed_bool
\@@_ot_validate_tag_aux:w 673, 674, 676	26, 56, 65, 69
\g_@@_pkg_euler_loaded_bool 2, 3, 8, 15	\l_@@_punctspace_adjust_tl
\c_@@_postadjust_tl	145, 150, 350, 355, 360, 662
150, 660	\g_@@_rawfeatures_sclist
\l_@@_postadjust_tl	58,
147, 367,	151, 275, 276, 482, 493, 621, 630, 656
377, 389, 499, 506, 534, 565, 568, 660	\@@_remove_clashing_featstr:n ..
\l_@@_pre_feat_sclist 152, 275, 493, 572 23, 71, 624
	\l_@@_rmfamily_family_tl 9, 10, 16, 153
	\@@_sanitise_fontname:Nn
 8, 9, 10, 44, 74, 75, 76, 84, 128
	\@@_save_family:nn
	33, 259
	\@@_save_family_needed:n
	220
	\@@_save_family_needed:nTF ...
	31, 220
	\@@_save_fontid_family:nn 236, 246, 258
	\@@_save_fontinfo:n
	261, 267
	\l_@@_saved_fontname_tl 105, 426, 439
	\l_@@_scale_tl
	106, 192, 272,
	276, 277, 291, 300, 484, 486, 491, 657

\l_@@_script_int	29, 42, 94, 142, 144, 150, 196, 200, 207, 277, 291	\g_@@_strong_prop	74, 116, 124, 125, 135, 143
\l_@@_script_name_tl	84, 131, 138, 139, 141, 181, 183, 184, 289, 301	\l_@@_strong_switch_tl ..	118, 143, 144
\l_@@_scriptlang_exist_bool	24, 283, 292, 297, 327, 335, 339	\l_@@_strong_tmp_tl	117, 135, 137
\@@_select_font_family:nn	1, 43, 150, 154, 157, 162	\l_@@_strongdef_int	39, 117, 124, 125, 126
\@@_set_autofont:Nnn	286, 287, 288, 293, 298, 301, 373	\@@_swap_plus_minus:n	72, 78
\@@_set_default_features:nn ..	60, 116	\@@_swap_plus_minus_aux:Nq ...	78, 79
\@@_set_faces:	264, 303	\l_@@_test_font	137, 143
\@@_set_faces_aux:nnnn	312, 314	\l_@@_tfm_bool	6, 343, 350
\@@_set_font_default_features:nnn	61, 121	\l_@@_this_feat_clist	60, 250, 258, 263
\@@_set_font_dimen:NnN ..	288, 289, 302	\l_@@_this_font_tl	109, 215, 216, 220, 248, 257, 263, 327, 333, 336
\@@_set_font_type:N	9, 27, 38, 64, 77, 90, 107, 120, 135, 143, 339	\l_@@_tmp_int ...	32, 429, 430, 438, 439
\@@_set_scriptlang:	27, 176	\l_@@_tmp_tl	41, 42, 44, 45, 93, 94, 110, 110, 111, 112, 113, 123, 124, 127, 128, 132, 135, 138, 139, 139, 140, 150, 167, 168, 171, 223, 224, 227, 228, 230, 234, 235, 236, 248, 250, 251, 253, 254, 255, 331
\@@_setboldmathrm_hook:nn ...	71, 91	\l_@@_tmpa_bool	19, 63, 66, 68
\@@_setmainfont_hook:nn	22, 85	\l_@@_tmpa_dim	42, 288, 293
\@@_setmathrm_hook:nn	65, 88	\l_@@_tmpa_font	391, 393
\@@_setmathsf_hook:nn	77, 89	\l_@@_tmpa_fp	40
\@@_setmathtt_hook:nn	83, 90	\l_@@_tmpa_int	33, 145, 147, 150, 155, 158
\@@_setmonofont_hook:nn	58, 87	\l_@@_tmpa_tl	28, 29, 53, 111
\@@_setsansfont_hook:nn	40, 86	\l_@@_tmpb_dim	43, 289, 294
\@@_setup_nfss:Nnn	448, 473, 477	\l_@@_tmpb_font	392, 393
\@@_setup_single_size:nn ...	428, 436	\l_@@_tmpb_fp	41
\l_@@_sffamily_family_tl	27, 28, 34, 154	\l_@@_tmpb_int	34, 143, 147, 155
\@@_shape_merge:nn	10, 11, 12, 13, 14, 15, 16, 17, 18, 22, 28, 33, 186, 511, 512	\l_@@_tmpb_tl	34, 39, 42, 46, 50, 53, 112, 132, 133, 134, 135
\g_@@_single_feat_tl	60, 76, 89, 265, 277, 279, 281, 293, 334, 351, 619	\l_@@_tmpc_dim	44
\l_@@_size_tl	107, 232, 438, 443, 444, 479, 491	\l_@@_tmpc_int	35, 149, 152
\l_@@_sizedfont_tl	108, 236, 439, 447, 449	\@@_trace:n	10
\l_@@_sizefeat_clist	48, 49, 209, 214, 330, 336	\l_@@_ttc_index_tl	95, 96, 100, 101, 113, 163, 641
\l_@@_sizing_leftover_clist	56, 442, 448, 474	\l_@@_ttfamily_family_tl	45, 46, 52, 155
\l_@@_smcp_shape_tl	120, 186, 189, 192, 195	\@@_update_featstr:n	19, 74, 172, 214, 218, 371, 466, 470, 482, 501, 509, 518, 575, 579, 614
\@@_strip_leading_sign:Nw ..	686, 689	\@@_warning:n	3, 4, 15, 29, 35, 94, 457, 461, 488, 526
\@@_strip_plus_minus:n	194, 196	\@@_warning:nn	5, 22, 63, 64, 68, 164, 218, 250, 302, 341, 381, 411, 422, 434
\@@_strip_plus_minus_aux:Nq	196, 197	\@@_warning:nnn	6, 34, 181, 191
\l_@@_strnum_int	31, 92, 98, 141, 152, 193, 208, 291, 333	\l_@@_wordspace_adjust_tl	146, 150, 328, 336, 661
\l_@@_strong_int	38, 134, 137, 138, 141, 143, 146, 157, 160	\\ ..	17, 25, 33, 33, 34, 37, 38, 39, 97, 98, 99, 157, 166, 171, 181, 182, 183, 204, 209, 215, 216, 217, 541, 553, 567, 568

\backslash	34	65, 66, 99, 111, 131, 153, 154, 166, 209, 211, 229, 269, 270, 292, 335, 350, 353, 357, 361, 365, 370, 465, 637 \backslash bool_until_do:nn 96, 147, 205 \backslash breve 23
A		C
\backslash acute 18		\backslash char 5, 6, 17
\backslash addfontfeature . . . 31, 48, 68, 96, 288, 295		char commands:
\backslash addfontfeatures 64, 76, 144		\backslash char_set_catcode_active:n . . . 28, 30
\backslash advance 74		\backslash char_set_catcode_ignore:n 220
\backslash aliasfontfeature		\backslash char_set_catcode_space:n 17
. 35, 88, 90, 201, 216, 230, 426		\backslash check 24
\backslash aliasfontfeatureoption		clist commands:
. 57, 58, 59, 92, 220, 355		\backslash clist_clear:N 103, 106, 272, 410
\backslash AtBeginDocument 43, 53, 112, 115, 127, 132		\backslash clist_count:N 251
\backslash author 36		\backslash clist_count:n 100
B		\backslash clist_gput_right:Nn 118
\backslash bar 22		\backslash clist_gset:Nn 1, 118
\backslash bfdefault 49, 76, 78,		\backslash clist_map_break: . . . 54, 66, 294, 336
81, 84, 88, 91, 92, 139, 140, 143, 306,		\backslash clist_map_inline:Nn . 48, 61, 206, 224
309, 310, 545, 548, 549, 557, 559, 561		\backslash clist_map_inline:nn 74, 74, 86, 121,
\backslash bfseries 161		124, 205, 224, 245, 284, 328, 428, 627
\backslash bgroup 37		\backslash clist_new:N
\backslash boldmath 28		46, 47, 48, 50, 51, 52, 53, 54, 55, 56,
bool commands:		57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67
\backslash bool_gset_false:N		\backslash clist_put_left:Nn 461
. . . 3, 6, 8, 10, 68, 98, 100, 101, 102,		\backslash clist_put_right:Nn
103, 104, 105, 106, 107, 112, 113,	 210, 536, 542, 544, 558, 564, 566
118, 119, 120, 121, 122, 123, 124, 125		\backslash clist_set:Nn
\backslash bool_gset_true:N 49, 99, 109, 175, 179, 186, 190,
. 2, 5, 7, 9, 13, 14, 15, 36, 62		194, 198, 204, 209, 214, 249, 254, 330
\backslash bool_if:NTF 1,		\backslash clist_set_eq:NN 250, 452
8, 10, 11, 17, 23, 23, 28, 29, 34, 35,		\backslash colon 30, 31, 116
39, 39, 40, 40, 58, 60, 65, 66, 68, 69,		\backslash convertcolorspec 401
72, 78, 78, 81, 84, 84, 88, 91, 97, 105,		cs commands:
108, 114, 121, 126, 134, 136, 161,		\backslash cs:w 127
173, 179, 185, 197, 202, 215, 217,		\backslash cs_end: 127
229, 249, 291, 296, 297, 320, 339,		\backslash cs_generate_variant:Nn
375, 410, 421, 433, 454, 471, 475,	 11, 12, 73, 77, 78, 79, 80,
480, 487, 533, 555, 575, 603, 606, 617		81, 82, 83, 84, 85, 86, 87, 88, 89, 90,
\backslash bool_if:nTF 87, 137, 188,		91, 92, 93, 155, 258, 435, 524, 669, 688
204, 284, 299, 305, 485, 527, 672, 691		\backslash cs_gset:Npn 29
\backslash bool_lazy_and:nnTF 27		\backslash cs_if_eq:NNTF 109, 166, 222
\backslash bool_new:N 1, 2, 3, 4,		\backslash cs_if_exist:NTF 3, 27, 192, 399
5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17,		\backslash cs_if_exist_p:N 30
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28		\backslash cs_new:Nn 1, 1, 1, 4, 5, 5,
\backslash bool_set_false:N		7, 7, 10, 11, 12, 13, 14, 15, 15, 19, 20,
. 18, 22, 29, 59, 61, 63,		25, 39, 40, 41, 43, 44, 47, 52, 52, 61,
65, 66, 95, 110, 114, 116, 134, 146,		67, 67, 70, 73, 74, 78, 79, 79, 94, 94,
169, 204, 204, 222, 271, 283, 327,		99, 112, 116, 118, 121, 132, 144, 146,
343, 344, 345, 346, 347, 538, 560, 636		151, 158, 158, 159, 160, 168, 168,
\backslash bool_set_true:N		
. 12, 26, 27, 28, 56, 57, 59,		

172, 175, 176, 185, 189, 196, 201, 220, 246, 252, 257, 259, 267, 278, 280, 282, 302, 303, 314, 319, 323, 325, 339, 353, 373, 397, 405, 421, 436, 477, 496, 501, 509, 515, 525, 537, 595, 596, 601, 614, 624, 654, 683	\DeclareUnicodeEncoding 21, 39
\cs_new:Npn 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 57, 65, 66, 79, 197, 234	\def 13, 28, 34, 42
\cs_new_eq:NN 56	\defaultfontfeatures 57
\cs_new_protected:Nn 1, 285, 292, 665, 710	\Delta 53
\cs_new_protected:Npn 58, 114	dim commands:
\cs_set:Npn 1, 5, 9, 31, 32, 33, 34, 50, 54, 60, 61, 66, 70, 72, 81, 88, 111, 112, 113, 152, 153, 174, 324, 416, 633, 670, 676, 689, 695	\dim_compare:nNnTF 305
\cs_set_eq:NN 31, 43, 60, 63, 85, 85, 86, 87, 88, 89, 90, 91, 92, 170, 175	\dim_eval:n 3, 7
\cs_to_str:N 101, 106, 111, 127, 168, 224	\dim_new:N 42, 43, 44
\cs_undefine:N 86, 242, 520	\dim_set:Nn 304
\cyrillicencoding 51, 55, 82	\dim_to_fp:n 293, 294
	dim internal commands:
	_dim_eval:w 76
	_dim_eval_end: 76
	\directlua 6, 113, 169, 225
	\discretionary 4
	\do 38
	\dospecials 38
	\dot 26
	\DTX 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
	E
	\else 8, 17, 36, 101, 211, 704, 705
	else commands:
	\else: 48
	\em 83, 110
	\emfontdeclare 58, 162
	\eminnershape 110, 162
	\emph 110
	\emreset 74, 106, 110
	\emshape 110, 162
	\EncodingAccent 15
	\EncodingCommand 9
	\EncodingComposite 27
	\EncodingCompositeCommand 33
	\encodingdefault 21, 39, 57, 80, 284
	\EncodingSymbol 21
	\endinput 8, 13, 22, 29
	environments:
	listingcont* 66
	verbatim* 50
	\ExecuteOptions 21
	exp commands:
	\exp_after:wN 441
	\exp_args:NNNx 298
	\exp_args:Nnnx 193
	\exp_args:Nnx 43, 44, 45, 49, 50
	\exp_args:No 17, 62, 89, 286, 330, 387, 598, 608
	\exp_args:NV 28
D	
\date 56	
\ddot 20	
\DeclareDocumentCommand 9, 15, 21, 27, 33, 39, 51, 64, 70, 76, 82	
\DeclareErrorFont 42	
\DeclareFontEncoding 30, 41	
\DeclareFontExtensions 104	
\DeclareFontFamily 44, 263	
\DeclareFontShape 46, 48, 50, 52, 53, 54, 54, 522	
\DeclareFontSubstitution 31, 43	
\DeclareMathAccent 18, 19, 20, 21, 22, 23, 24, 25, 26, 27	
\DeclareMathDelimiter ... 65, 66, 67, 68, 69	
\DeclareMathSymbol 31, 36, 37, 38, 39, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 70	
\DeclareOption ... 1, 5, 6, 7, 8, 9, 10, 11, 16	
\DeclareRobustCommand 2, 4, 13, 31, 38, 43, 48, 49, 53, 83, 130	
\DeclareSymbolFont 16, 74	
\DeclareSymbolFontAlphabet 76	
\DeclareTextCommand 7, 13, 19	
\DeclareTextComposite 31	
\DeclareTextCompositeCommand 37	
\DeclareTextFontCommand 9, 110, 151	
\DeclareTextSymbol 25	
\DeclareUnicodeAccent 6	

\exp_args:Nx	64, 71, 428	\l_fontspec_feature_string_tl	19, 53
\exp_args:Nxx	178	\fontspec_font_if_exist:n	166
\exp_last_unbraced:Nn	31, 32, 33, 34	\fontspec_font_if_exist:nTF	175
\exp_not:N	15, 16, 17, 20, 33, 34, 35, 51, 52, 53, 83, 104, 106, 107, 108, 234, 235, 238, 239, 247, 248, 276, 277, 407, 542, 554, 567, 568	\l_fontspec_fontname_tl 8, 18, 21, 42, 45, 80, 102, 115, 120, 125, 126, 130, 140, 192, 274, 288, 293, 298, 302, 305, 392, 401, 418, 423, 426, 447, 468, 479, 488, 492, 537, 545, 559, 567
\exp_not:n 13, 31, 49, 54, 60, 191, 260, 541, 553	\fontspec_if_aat_feature:nn	5
\expandafter	29	\fontspec_if_aat_feature:nnTF	5
F			
\familydefault	20, 38, 56	\fontspec_if_current_feature:n	176
\FancyVerbSpace	85	\fontspec_if_current_feature:nTF 176, 281
\fi	10, 19, 29, 32, 36, 98, 103, 110, 116, 213, 358, 367, 704, 705	\fontspec_if_current_language:n	131
fi commands:		\fontspec_if_current_language:nTF 131
\fi:	50	\fontspec_if_current_script:n	116
file commands:		\fontspec_if_current_script:nTF	116
\file_if_exist:nTF	25, 89	\fontspec_if_feature:n	34
\file_input:n	90	\fontspec_if_feature:nnn	60
\filedate	56	\fontspec_if_feature:nnnTF	60
\fileversion	56	\fontspec_if_feature:nTF	34
\fmtname	29	\fontspec_if_fontspec_font:	1
\font	3, 6, 7, 9, 9, 12, 16, 27, 38, 42, 50, 64, 67, 77, 80, 90, 97, 107, 109, 110, 120, 135, 166, 222, 288, 309, 330, 331, 332, 338, 339, 340, 351, 356, 361, 378, 390	\fontspec_if_fontspec_font:TF	1, 7, 25, 36, 62, 75, 88, 105, 118, 133, 147
font commands:		\fontspec_if_language:n	86
\l_fontspec_font	42, 148, 153, 157, 164	\fontspec_if_language:nn	103
\fontdimen	81, 82, 304, 330, 331, 332, 338, 339, 340, 351, 356, 361	\fontspec_if_language:nnTF	103
\fontdimen8	80	\fontspec_if_language:nTF	86
\fontencoding	4, 15, 33, 51, 107, 284	\fontspec_if_opentype:	23
\fontfamily	16, 29, 34, 52, 106, 161, 285	\fontspec_if_opentype:TF	23
\fontname	40, 56	\fontspec_if_script:n	73
\fontshape	7, 22, 23	\fontspec_if_script:nTF	73
\fontspec	1, 24, 31, 42, 127	\fontspec_if_small_caps:	182
fontspec commands:		\fontspec_if_small_caps:TF	182
\fontspec_calc_scale:n	79	\l_fontspec_lang_tl 48, 143, 180, 280, 332, 349, 580, 591
\fontspec_complete_fontname:Nn 105, 115, 120, 135, 156, 160, 170, 236, 316, 319	\fontspec_maybe_setup_maths:	94
\g_fontspec_default_fontopts_tl	31	\fontspec_merge_shape:n 19, 41, 46, 51, 56
\g_fontspec_encoding_tl 23, 41, 42, 44, 48, 49, 51, 52, 55, 56, 74, 75, 77, 78, 78, 79, 80, 81, 84, 87, 88, 89, 91, 92, 645	\l_fontspec_mode_tl 71, 75, 76, 84, 587, 647
\l_fontspec_family_tl 41, 42, 77, 152, 163	\fontspec_new_lang:nn	102, 323
		\fontspec_new_script:nn	98, 278
		\fontspec_parse_colour:niii	408, 416
		\fontspec_parse_cv:w	234, 247
		_fontspec_parse_wordspace:w	321, 324
		\fontspec_patch_fancyvrb:	47, 81
		\fontspec_patch_listings:	48, 88

`\fontspec_patch_moreverb:` ... 46, 66
`\fontspec_patch_verbatim:` ... 45, 50
`\fontspec_print_visible_spaces:`
 27, 42, 56, 62, 77
`\l_fontspec_renderer_tl` 52,
 55, 59, 60, 79, 164, 354, 362, 366, 642
`\l_fontspec_script_tl` . 47, 95, 129,
 140, 180, 279, 290, 577, 579, 588, 590
`\fontspec_select:nn` 43
`\fontspec_set_em_level:n` 159
`\fontspec_set_family:Nnn` . 3, 9, 27,
 45, 63, 64, 69, 70, 75, 76, 81, 82, 101, 146
`\fontspec_set_fontface:NNnn` 156
`\fontspec_set_strong_level:n` ... 160
`\fontspec_setup_maths:` 1, 129
`\fontspec_tmp:` 60, 63
`\fontspec_visible_space:` 14, 31, 85, 92
`\fontspec_visible_space_fallback:`
 18, 20
fontspec internal commands:
`\l_fontspec_check_bool`
 ... 95, 99, 105, 114, 146, 154, 161, 173
`_fontspec_update_featstr:n` 97
`\FONTSPECCTX` 2
`\FontspecSetCheckBoolFalse` 65
`\FontspecSetCheckBoolTrue` 65
fp commands:
`\fp_eval:n` 277, 293
`\fp_new:N` 40, 41

G

`\g` 117
`\Gamma` 52
`\gdef` 2
`\GetFileInfo` 55
`\global` 7, 74
`\grave` 19
group commands:

`\group_begin:`
 4, 27, 28, 37, 56, 72, 119, 149,
 168, 266, 282, 287, 294, 390, 399, 518
`\group_end:` 33,
 33, 39, 41, 42, 81, 128, 160, 172, 173,
 274, 290, 297, 299, 394, 395, 403, 521

H

`\hat` 25, 115
`\hbar` 71
`\hbox` 36
`\hyphenchar` 6, 9

I

`\IfBooleanTF` 118, 130
`\ifcase` 348
`\IfFontExistsTF` 175
`\IfFontFeatureActiveTF` 108, 257
`\ifmmode` 36
`\IfNoValueTF` 59
`\ifnum` 6, 98, 207, 355
`\ifx` 15, 29, 30, 98, 110, 116, 704, 705
`\ignorespaces` 4, 9, 14, 19, 62, 166
`\InputIfFileExists` 3

int commands:

`\int_case:nn` 57, 73
`\int_case:nnTF` 310
`\int_compare:nNnTF` 152, 712
`\int_compare:nTF`
 32, 52, 100, 251, 404, 407, 438, 678, 697
`\int_compare_p:nNn` 96, 147, 205
`\int_eval:n` 251
`\int_if_even:nTF` 37
`\int_incr:N`
 79, 100, 102, 126, 141, 158, 212
`\int_new:N`
 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39
`\int_set:Nn` ... 11, 42, 45, 76, 78, 93,
 94, 96, 100, 137, 143, 149, 155, 159,
 160, 197, 210, 291, 333, 429, 648, 700
`\int_to_hex:n` 439
`\int_use:N`
 78, 93, 97, 102, 125, 134, 138, 143
`\int_zero:N` 61, 94, 105, 117, 145, 146,
 156, 157, 203, 350, 649, 650, 651, 714
`\l_tmpa_int` 94,
 96, 98, 100, 102, 203, 205, 208, 210, 212
`\l_tmpb_int` .. 93, 96, 100, 197, 205, 210
`\itdefault` 2, 11, 13, 15, 41, 77,
 84, 89, 307, 309, 529, 530, 534, 546, 548
`\itscdefault` 2, 7, 11, 13, 15, 16, 17, 558, 559
`\itshape` 38, 66, 112

K

keys commands:

`\l_keys_choice_int` 52, 57, 73
`\keys_define:nn`
 . 3, 3, 7, 9, 20, 20, 22, 28, 47, 69, 81,
 92, 170, 199, 210, 212, 218, 230, 237,
 241, 244, 254, 264, 273, 280, 318,
 325, 416, 497, 505, 514, 524, 529, 551
`\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

<code>\keys_set:nn</code>	8, 13, 32, 42, 79, 83, 88, 184, 185, 200, 205, 210, 213, 231, 238, 245, 312, 342, 459
<code>\keys_set_groups:nnn</code>	411
<code>\keys_set_known:nn</code>	15
<code>\keys_set_known:nnN</code>	70, 80, 150, 441
<code>\l_keys_value_tl</code>	114, 119, 124, 129
L	
<code>\l</code>	31, 51, 54, 64, 65, 66, 72
<code>\Lambda</code>	55
<code>\latinencoding</code>	52, 56, 83
<code>\leavevmode</code>	36
<code>\let</code>	38
<code>\liningnums</code>	2, 35, 119, 285
<code>listingcont*</code> (environment)	66
<code>\LuaLaTeX</code>	34
M	
<code>\mathalpha</code>	18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62
<code>\mathbf</code>	56, 78, 88, 117
<code>\mathbin</code>	63
<code>\mathchar</code>	71
<code>\mathchardef</code>	29
<code>\mathclose</code>	36, 39, 66, 68
<code>\mathdollar</code>	70
<code>\mathit</code>	40, 56, 77, 84, 89
<code>\mathopen</code>	65, 67
<code>\mathord</code>	69, 70
<code>\mathpunct</code>	31, 38
<code>\mathrel</code>	37, 64
<code>\mathring</code>	27
<code>\mathrm</code>	28, 76, 87, 117
<code>\mathsf</code>	79, 91
<code>\mathtt</code>	80, 92
<code>\mddefault</code>	74, 75, 77, 79, 80, 87, 89, 305, 307, 308, 544, 546, 547, 556, 558, 560
<code>\mkern</code>	71
msg commands:	
<code>\msg_error:nn</code>	1
<code>\msg_error:nnn</code>	2, 3
<code>\msg_fatal:nn</code>	21
<code>\msg_info:nn</code>	7
<code>\msg_info:nnn</code>	8
<code>\msg_info:nnnn</code>	9
<code>\msg_line_context:</code>	96
<code>\msg_new:nnn</code>	11, 14, 15
<code>\msg_new:nnnn</code>	12, 16
<code>\msg_redirect_module:nnn</code>	13, 14, 18, 19
<code>\msg_redirect_name:nnn</code>	458
<code>\msg_trace:nn</code>	10
<code>\msg_warning:nn</code>	4
<code>\msg_warning:nnn</code>	5
<code>\msg_warning:nnnn</code>	6
N	
<code>\newAATfeature</code>	76, 175
<code>\NewDocumentCommand</code>	1, 6, 11, 16, 21, 25, 29, 33, 37, 41, 43, 45, 49, 53, 57, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100, 104, 108, 119
<code>\newfontface</code>	30, 53
<code>\newfontfamily</code>	41, 114
<code>\newfontfeature</code>	32, 32, 32, 72, 168
<code>\newfontlanguage</code>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 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, 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>	84
<code>\newopentypefeature</code>	80, 185
<code>\normalfont</code>	23, 41, 59, 73, 120, 155
<code>\normalsize</code>	50, 519
<code>\null</code>	36
<code>\nullfont</code>	15
<code>\numexpr</code>	44
O	
<code>\oldstylenums</code>	2, 35, 112, 285
<code>\Omega</code>	62
<code>\or</code>	351, 359, 363
P	
<code>\par</code>	75
<code>Path</code>	24
<code>\Phi</code>	60
<code>\Pi</code>	57
<code>\postexhyphenchar</code>	651
<code>\posthyphenchar</code>	649
<code>\preexhyphenchar</code>	650
<code>\prehyphenchar</code>	648
prg commands:	
<code>\prg_new_conditional:Nnn</code> 1, 5, 23, 25, 26, 34, 44, 58, 60, 73, 81, 86, 103, 116, 127, 131, 131, 166, 176, 178, 182, 183, 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, 84, 88, 97, 99, 101, 105, 110, 112, 114, 121, 125, 127, 129, 129, 138, 140, 142, 144, 161, 173, 173, 180, 181, 189, 197, 200, 215, 229, 231, 279, 282, 394
<code>\prg_return_true:</code>	3, 13, 16, 24, 28, 29, 36, 42, 47, 50, 54, 67, 68, 80, 85, 97, 105, 110, 117, 125, 129, 135, 140, 161, 172, 173, 180, 181, 186, 198, 215, 229, 237, 243, 282, 395
<code>\prg_set_conditional:Nnn</code>	13, 21, 26, 35
<code>\ProcessOptions</code>	22
prop commands:	
<code>\prop_gclear:N</code>	60, 116
<code>\prop_get:NnN</code>	41, 44, 47, 48, 93, 95, 123, 138, 151, 152
<code>\prop_get:NnNTF</code>	85, 86, 94, 102, 102, 105, 132, 135, 143, 228, 248, 409
<code>\prop_gput:Nnn</code>	11, 78, 84, 125, 135, 140, 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, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355
<code>\prop_gput_if_new:Nnn</code>	77, 83, 124
<code>\prop_gremove:Nn</code>	139
<code>\prop_if_in:NnTF</code>	9, 87
<code>\prop_map_inline:Nn</code>	312

\prop_new:N		\settoheight	307
.. 68, 69, 70, 71, 72, 73, 74, 75, 76, 269		\sfdefault	28, 38, 46, 97, 126
\prop_put:Nnn	81, 82, 148, 181, 335	\sffamily	31
\providecommand	1, 2, 2, 3, 3, 4, 5, 6	\Sigma	58
\ProvidesExplFile	49	\sishape	<u>2</u>
\ProvidesExplPackage	44, 45, 46	\sldefault	3, 12,
\Psi	61	14, 16, 46, 308, 310, 530, 533, 547, 549	
Q			
\quad	56	\slscdefault ..	3, 12, 14, 15, 16, 18, 560, 561
quark commands:		\slshape	<u>38</u> , 64
\q_nil		\space	34, 39, 44, 192
78, 79, 196, 197, 235, 248, 667, 670,		str commands:	
673, 674, 676, 686, 689, 692, 693, 695		\c_backslash_str	87
\q_stop	321, 324	\c_colon_str	235, 248
R			
\relax ...	5, 6, 29, 36, 44, 45, 50, 55, 74, 355	\str_case:nn	80, 542, 554
\renewcommand	71	\str_case:nnTF	199, 267
\RenewDocumentCommand	47, 114	\str_case_e:nnTF	346
\renewfontfamily	45	\str_if_eq:eeTF	
\RequirePackage	5, 43, 48, 48, 62	20, 38, 56, 72, 93, 156, 216, 393
\RequirePackageWithOptions	7, 12	\str_if_eq:nnTF	
\rmdefault ..	10, 20, 29, 45, 96, 117, 125, 285	87, 124, 139, 309, 375, 455
\rmfamily	13, 80, 309	\str_if_eq_p:ee	
S			
scan commands:		299, 307, 308, 309, 529, 530
\scan_stop:	3, 7, 17, 46, 54	\str_if_eq_p:nn	672, 691
\scdefault	2, 3, 11, 12,	\str_lower_case:n	72, 93
13, 14, 17, 18, 51, 511, 512, 513, 556, 557		\string	21, 56, 76, 96
\scshape	<u>38</u>	\strong	<u>151</u>
\select	19	\strongenv	<u>130</u> , <u>151</u>
\selectfont		\strongfontdeclare	<u>114</u> , <u>161</u>
.. 5, 7, 17, 22, 23, 35, 53, 108, 161, 286		\strongreset	<u>121</u> , <u>147</u> , <u>151</u>
seq commands:		\suppressfontnotfounderror	11
\seq_if_empty:NTF	137	sys commands:	
\seq_new:N	45	\sys_if_engine luatex:TF	3
\seq_put_right:Nn	125, 140	\sys_if_engine xetex:TF	10
\setboldmathrm	25, 28, <u>67</u> , 93, 117	T	
\setfontfamily	49	TeX and L ^A T _E X 2 _ε commands:	
\setmainfont	6, <u>7</u> , 24, 28, 115		
\SetMathAlphabet		\@	31, 60
... 77, 78, 79, 80, 84, 87, 88, 89, 91, 92		\@sverb	40, 42
\setmathrm	21, <u>61</u> , 92, 117	\@filelist	50
\setmathsf	29, <u>73</u> , 94	\@ifpackageloaded	
\setmathtt	33, <u>79</u> , 95	1, 6, 13, 14, 15, 52, 68,
\setmonofont	16, <u>43</u>	83, 90, 96, 100, 101, 102, 103, 104,	
\setromanfont	<u>37</u>	105, 106, 107, 108, 112, 113, 114,	
\setsansfont	11, <u>25</u>	118, 119, 120, 121, 122, 123, 124, 125	
\SetSymbolFont	17, 75, 81	\@ifstar	40
		\@makeother	38
		\@noligs	39
		\@nomath	85, 132
		\@onlypreamble	92, 93, 94, 95
		\@sverb	42
		\@sxverbatim	62

<code>\@tempa</code>	29, 30	<code>\tl_count:n</code>	404, 407, 678, 697
<code>\@verb</code>	40	<code>\tl_gclear:N</code>	265, 643, 644, 656
<code>\@verbatim</code>	56, 62, 77	<code>\tl_gput_right:Nn</code>	539, 621
<code>\add@unicode@accent</code>	5, 7, 19	<code>\tl_gremove_all:Nn</code>	630
<code>\color@</code>	399	<code>\tl_gset:Nn</code>	41, 42,
<code>\curr@fontshape</code>	110, 167, 223		44, 60, 76, 96, 97, 98, 124, 125, 126,
<code>\define@antt@mathversions</code>	98		127, 139, 240, 255, 293, 334, 351, 619
<code>\define@iwona@mathversions</code>	110	<code>\tl_gset_eq:NN</code>	230, 241, 645
<code>\define@kurier@mathversions</code>	116	<code>\tl_if_empty:NTF</code>	
<code>\f@encoding</code>	32, 192, 195, 196		.. 29, 46, 50, 82, 181, 213, 215, 226,
<code>\f@family</code> ...	3, 3, 32, 41, 44, 47, 48,		276, 279, 333, 354, 362, 366, 379,
	93, 95, 123, 138, 151, 152, 192, 195, 196		443, 456, 484, 503, 540, 562, 577, 588
<code>\f@series</code>		<code>\tl_if_empty:nTF</code> 7, 14, 18, 59, 89, 90,	
	.. 23, 32, 124, 135, 138, 192, 195, 196		91, 109, 129, 138, 164, 326, 329, 377, 565
<code>\f@shape</code>	22, 23, 28, 33, 77, 86, 186	<code>\tl_if_empty_p:e</code>	87, 137, 188
<code>\f@size</code>	39, 110,	<code>\tl_if_empty_p:n</code>	88
	139, 146, 167, 223, 391, 392, 413, 520	<code>\tl_if_eq:NNTF</code>	194, 419, 431
<code>\listing@line</code>	74	<code>\tl_if_eq:nnTF</code>	92, 143
<code>\lst@visiblespace</code>	92	<code>\tl_if_exist:NTF</code>	151, 511
<code>\not@math@alphabet</code> ...	6, 40, 45, 50, 55	<code>\tl_if_exist_p:N</code>	28
<code>\reset@font</code>	153	<code>\tl_if_in:NnTF</code>	50, 50, 254
<code>\thelisting@line</code>	74	<code>\tl_if_in:nnTF</code>	64, 65, 66, 178
<code>\two@digits</code>	226, 238, 239	<code>\tl_if_single:nTF</code>	126, 383
<code>\verb@eol@error</code>	38	<code>\tl_new:N</code>	
<code>\verbatim@font</code>	39		.. 77, 78, 79, 80, 84, 85, 86, 87, 88,
<code>\verbatim@line</code>	75		89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
<code>\verbatim@processline</code>	72		99, 100, 101, 102, 103, 104, 105, 106,
<code>\verbatim@start</code>	56, 77		107, 108, 109, 110, 111, 112, 113,
<code>\xlx@defaultthyphenchar</code>	7, 13		114, 115, 116, 117, 118, 119, 120,
<code>\z@</code>	6		121, 122, 123, 124, 128, 129, 130,
tex commands:			131, 132, 133, 134, 135, 136, 137,
<code>\tex_font:D</code>	59		138, 140, 141, 143, 144, 145, 146,
<code>\tex_hyphenchar:D</code>	54		147, 151, 151, 152, 153, 154, 155, 156
<code>\tex_iffontchar:D</code>	46	<code>\tl_put_left:Nn</code>	46
<code>\textsc</code>	37	<code>\tl_put_right:Nn</code> 134, 367, 377, 389, 489	
<code>\textsf</code>	33	<code>\tl_remove_all:Nn</code> ...	47, 86, 235, 323
<code>\textsi</code>	2	<code>\tl_remove_once:Nn</code>	52
<code>\textvisiblespace</code>	24	<code>\tl_replace_all:Nnn</code>	90, 93, 322
<code>\the</code>	75	<code>\tl_set:Nn</code>	
<code>\Theta</code>	54		.. 21, 22, 26, 28, 29, 34, 39, 39, 42,
<code>\tilde</code>	21		45, 46, 46, 47, 53, 53, 55, 58, 71, 84,
<code>\title</code>	32		85, 86, 89, 95, 96, 100, 101, 110, 111,
tl commands:			127, 139, 142, 148, 149, 153, 157,
<code>\c_empty_tl</code>			160, 161, 164, 167, 168, 189, 216,
	.. 60, 673, 674, 692, 693, 704, 705		220, 223, 224, 232, 234, 244, 250,
<code>\tl_clear:N</code>	24,		253, 272, 276, 277, 289, 290, 291,
	45, 107, 133, 248, 258, 301, 424, 425,		300, 321, 327, 328, 332, 336, 349,
	438, 639, 640, 641, 642, 657, 661, 662		350, 354, 355, 360, 362, 366, 382,
<code>\tl_clear_new:N</code>	81, 82, 83		384, 385, 405, 418, 424, 436, 445,
<code>\tl_const:Nn</code>	11,		449, 477, 486, 494, 535, 557, 572, 647
	12, 13, 14, 15, 16, 17, 18, 148, 149, 150		

<code>\tl_set_eq:NN</code>	10, 21, 28, 39, 41, 46, 49, 51, 52, 55, 56, 57, 57, 62, 126, 145, 152, 163, 186, 257, 426, 439, 537, 545, 559, 567, 658, 659, 660
<code>\tl_to_str:N</code>	21
<code>\tl_to_str:n</code>	87, 179
<code>\tl_trim_spaces:n</code>	14, 16
<code>\tl_use:N</code>	22, 33, 512
<code>\tmpa</code>	28, 29
token commands:	
<code>\token_to_str:N</code>	87, 399
<code>\ttdefault</code>	46, 47, 56, 98, 127
<code>\ttfamily</code>	49
<code>\typeout</code>	3, 5, 23, 32, 37, 54, 59, 60, 69, 71, 75, 83, 83, 91, 93, 96, 97, 112, 116, 120, 120, 123, 133, 134, 134, 136, 138, 142, 146, 150, 154, 178, 183, 191, 192, 199, 203, 203, 209, 210, 218, 223, 224, 228, 236, 243, 253, 256, 259, 260, 276, 277, 282, 288, 341, 349, 352, 356, 360, 364, 407, 423, 444, 449, 460, 464, 479, 482, 517, 616, 620, 626, 629, 635, 685
U	
<code>\UndeclareAccent</code>	<u>64</u>
<code>\UndeclareCommand</code>	<u>64</u>
<code>\UndeclareComposite</code>	<u>82</u>
<code>\UndeclareSymbol</code>	<u>64</u>
<code>\UndeclareTextCommand</code>	68, 74, 80
<code>\unexpanded</code>	96, 123
<code>\UnicodeEncodingName</code>	13, 19, 25, 31, 37, 57, 58, 62, 68, 74, 80, 87
<code>\UnicodeFontFile</code>	1, 47, 49, 51, 53, 55
<code>\UnicodeFontName</code>	2
<code>\UnicodeFontTeXLigatures</code>	3, 4, 47, 49, 51, 53, 55
<code>\updefault</code>	17, 18, 56, 74, 75, 78, 79, 80, 81, 87, 88, 91, 92, 196, 305, 306, 544, 545
<code>\upshape</code>	<u>38</u> , 111, 113
<code>\Upsilon</code>	59
<code>\url</code>	40
use commands:	
<code>\use:N</code>	99, 106
<code>\use:n</code>	11, 29, 47, 102, 155, 170, 232, 441
<code>\use_i:nnn</code>	263
<code>\use_ii:nnn</code>	263
<code>\use_iii:nnn</code>	249
<code>\use_none:nn</code>	85, 86, 87, 88, 89, 90, 91
<code>\usefont</code>	23
<code>\UTFencname</code>	49, 81
V	
<code>\verb</code>	<u>34</u> , 127
<code>\verb*</code>	<u>34</u> , 126
<code>verbatim* (environment)</code>	<u>50</u>
X	
<code>\XeLaTeX</code>	34
<code>\XeTeXcountvariations</code>	355
<code>\XeTeXfeaturename</code>	28
<code>\XeTeXfonttype</code>	348
<code>\XeTeXisexclusivefeature</code>	32
<code>\XeTeXOTcountfeatures</code>	199
<code>\XeTeXOTcountlanguages</code>	144
<code>\XeTeXOTcountscripts</code>	93
<code>\XeTeXOTfeaturetag</code>	207
<code>\XeTeXOTlanguagetag</code>	150
<code>\XeTeXOTscripttag</code>	98
<code>\XeTeXpicfile</code>	60, 61, 63
<code>\XeTeXselectorname</code>	34, 39, 44
<code>\Xi</code>	56