

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/>

2019/01/18 v2.6l

Contents

I	fontspec.dtx	6
1	Package declaration	6
1.1	Lua header	7
II	fontspec-code-load.dtx	8
1	The fontspec.sty loading file	8
III	fontspec-code-vars.dtx	9
1	Declaration of variables	9
IV	fontspec-code-msg.dtx	14
1	Error/warning/info messages	14
1.1	Errors	14
1.2	Warnings	15
1.3	Info messages	17
V	fontspec-code-opening.dtx	19
1	Opening code	19
1.1	Package options	19
1.2	Encodings	19

I.3	Generic functions	20
I.4	expl3 variants	21
VI	fontspec-code-fontload.dtx	22
1	expl3 interface for primitive font loading	22
VII	fontspec-code-interfaces.dtx	24
1	User commands	24
VIII	fontspec-code-user.dtx	27
1	User command internals	27
I.1	Font selection	27
I.2	Font feature selection	30
I.3	Defining new font features	32
IX	fontspec-code-api.dtx	35
1	Programmer's interface	35
X	fontspec-code-internal.dtx	41
1	Internals	41
I.1	The main function for setting fonts	41
I.2	Setting font shapes in a family	49
I.3	Initialisation	58
I.4	Miscellaneous	59
XI	fontspec-code-opentype.dtx	61
1	OpenType definitions code	61
I.1	Adding features when loading fonts	62
I.2	OpenType feature information	66
XII	fontspec-code-graphite.dtx	69
1	Graphite/AAT code	69
XIII	fontspec-code-keyval.dtx	71
1	Font loading (keyval) definitions	71

1.1	Pre-pre-parsing stages	71
1.2	Pre-parsed features	73
1.3	Font faces	73
1.4	General font-independent features	76
XIV fontspec-code-feat-opentype.dtx		86
1	OpenType feature definitions	86
2	Regular key=val / tag definitions	86
2.1	Ligatures	86
2.2	Letters	86
2.3	Numbers	87
2.4	Vertical position	87
2.5	Contextuals	87
2.6	Diacritics	88
2.7	Kerning	88
2.8	Fractions	88
2.9	Style	89
2.10	CJK shape	89
2.11	Character width	90
2.12	Vertical	90
3	OpenType features that need numbering	90
3.1	Alternate	90
3.2	Variant / StylisticSet	91
3.3	CharacterVariant	91
3.4	Annotation	92
3.5	Ornament	92
4	Script and Language	92
4.1	Script	92
4.2	Language	93
5	Backwards compatibility	94
XV fontspec-code-scripts.dtx		95
1	Font script definitions	95
XVI fontspec-code-lang.dtx		98
1	Font language definitions	98

XVII	fontspec-code-feat-aat.dtx	106
1	AAT feature definitions	106
1.1	Ligatures	106
1.2	Letters	106
1.3	Numbers	107
1.4	Contextuals	107
1.5	Diacritics	107
1.6	Vertical position	107
1.7	Fractions	107
1.8	Alternate	107
1.9	Variant / StylisticSet	108
1.10	Style	108
1.11	CJK shape	108
1.12	Character width	109
1.13	Annotation	109
XVIII	fontspec-code-enc.dtx	110
1	Extended font encodings	110
XIX	fontspec-code-math.dtx	113
1	Selecting maths fonts	113
XX	fontspec-code-closing.dtx	118
1	Closing code	118
1.1	Finishing up	118
XXI	fontspec-code-xfss.dtx	119
1	Changes to the NFSS	119
1.1	Italic small caps and so on	119
1.2	Emphasis	120
1.3	Strong emphasis	122
XXII	fontspec-code-patches.dtx	124
1	Patching code	124
1.1	\-	124
1.2	Verbatims	124
1.3	\oldstylenums	126

File I

fontspec.dtx

1 Package declaration

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

```
1 <*dtx>
2 \gdef\FONTSPECDTX{
3   \DTX{fontspec.dtx}
4   \DTX{fontspec-code-load.dtx}
5   \DTX{fontspec-code-vars.dtx}
6   \DTX{fontspec-code-msg.dtx}
7   \DTX{fontspec-code-opening.dtx}
8   \DTX{fontspec-code-fontload.dtx}
9   \DTX{fontspec-code-interfaces.dtx}
10  \DTX{fontspec-code-user.dtx}
11  \DTX{fontspec-code-api.dtx}
12  \DTX{fontspec-code-internal.dtx}
13  \DTX{fontspec-code-opentype.dtx}
14  \DTX{fontspec-code-graphite.dtx}
15  \DTX{fontspec-code-keyval.dtx}
16  \DTX{fontspec-code-feat-opentype.dtx}
17  \DTX{fontspec-code-scripts.dtx}
18  \DTX{fontspec-code-lang.dtx}
19  \DTX{fontspec-code-feat-aat.dtx}
20  \DTX{fontspec-code-enc.dtx}
21  \DTX{fontspec-code-math.dtx}
22  \DTX{fontspec-code-closing.dtx}
23  \DTX{fontspec-code-xfss.dtx}
24  \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 {2019/01/18}{2.61}{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 = "2.61",
63 <lua>    date = "2019/01/18",
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 ??.)

`\l_@@_scripts_missing_bool` True for a regular opentype font with at least the DFLT script. False for a font lacking all script (and therefore language and feature) tags.

```
29 \bool_new:N \l_@@_scripts_missing_bool
```

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

Counters

```
30 \int_new:N \l_@@_script_int
```

```
31 \int_new:N \l_@@_language_int
```

```
32 \int_new:N \l_@@_strnum_int
```

```
33 \int_new:N \l_@@_tmp_int
```

```
34 \int_new:N \l_@@_tmpa_int
```

```
35 \int_new:N \l_@@_tmpb_int
```

```
36 \int_new:N \l_@@_tmpc_int
```

```
37 \int_new:N \l_@@_em_int
```

```
38 \int_new:N \l_@@_emdef_int
```

```
39 \int_new:N \l_@@_strong_int
```

```
40 \int_new:N \l_@@_strongdef_int
```

Floats

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

```
42 \fp_new:N \l_@@_tmpb_fp
```

Dimensions

```
43 \dim_new:N \l_@@_tmpa_dim
```

```
44 \dim_new:N \l_@@_tmpb_dim
```

```
45 \dim_new:N \l_@@_tmpc_dim
```

Sequences

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

Comma-lists

```
47 \clist_new:N \g_@@_default_fontopts_clist
```

```
48 \clist_new:N \g_@@_all_keyval_modules_clist
```

```
49 \clist_new:N \l_@@_sizefeat_clist
```

```
50 \clist_set:Nn \l_@@_sizefeat_clist {Size={-}}
```

```
51 \clist_new:N \l_@@_extensions_clist
```

```
52 \clist_new:N \l_@@_fontopts_clist
```

```
53 \clist_new:N \l_@@_family_fontopts_clist
```

```
54 \clist_new:N \l_@@_all_features_clist
```

```
55 \clist_new:N \l_@@_leftover_clist
```

```
56 \clist_new:N \l_@@_keys_leftover_clist
```

```

57 \clist_new:N \l_@@_sizing_leftover_clist
58 \clist_new:N \l_@@_fontfeat_clist
59 \clist_new:N \l_@@_fontfeat_curr_clist
60 \clist_new:N \l_@@_arg_clist
61 \clist_new:N \l_@@_this_feat_clist

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

```

Property lists

```

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

```

Token lists

```

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

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

85 \tl_new:N \l_fontspec_mode_tl
86 \tl_new:N \g_@@_curr_series_tl
87 \tl_new:N \g_@@_defined_shapes_tl
88 \tl_new:N \g_@@_nfss_enc_tl
89 \tl_new:N \g_@@_nfss_family_tl
90 \tl_new:N \g_@@_single_feat_tl
91 \tl_new:N \l_@@_basename_tl
92 \tl_new:N \l_@@_curr_fontname_tl
93 \tl_new:N \l_@@_curr_bfname_tl
94 \tl_new:N \l_@@_ext_filename_tl
95 \tl_new:N \l_@@_extension_tl
96 \tl_new:N \l_@@_font_path_tl
97 \tl_new:N \l_@@_fontid_tl
98 \tl_new:N \l_@@_fontname_tl
99 \tl_new:N \l_@@_hexcol_tl
100 \tl_new:N \l_@@_nfss_sc_tl
101 \tl_new:N \l_@@_nfss_tl

```

```

102 \tl_new:N \l_@@_nfss_fam_tl
103 \tl_new:N \l_@@_opacity_tl
104 \tl_new:N \l_@@_optical_size_tl
105 \tl_new:N \l_@@_options_tl
106 \tl_new:N \l_@@_saved_fontname_tl
107 \tl_new:N \l_@@_scale_tl
108 \tl_new:N \l_@@_size_tl
109 \tl_new:N \l_@@_sizedfont_tl
110 \tl_new:N \l_@@_this_font_tl
111 \tl_new:N \l_@@_tmp_tl
112 \tl_new:N \l_@@_tmpa_tl
113 \tl_new:N \l_@@_tmpb_tl
114 \tl_new:N \l_@@_ttc_index_tl
115 \tl_new:N \l_@@_emshape_query_tl
116 \tl_new:N \l_@@_em_switch_tl
117 \tl_new:N \l_@@_em_tmp_tl
118 \tl_new:N \l_@@_strong_tmp_tl
119 \tl_new:N \l_@@_strong_switch_tl
120 \tl_new:N \l_@@_hyphenchar_tl
121 \tl_new:N \l_@@_smcp_shape_tl

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

    Defaults:

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

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

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

139 \tl_new:N \l_@@_script_name_tl
140 \tl_new:N \l_fontspect_script_tl
141 \tl_new:N \l_@@_lang_name_tl
142 \tl_new:N \l_fontspect_lang_tl

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

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

150 \tl_new:N \g_@@_rawfeatures_sclist

151 \tl_new:N \l_@@_pre_feat_sclist

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

152 \tl_new:N \l_@@_rmfamily_family_tl

153 \tl_new:N \l_@@_sffamily_family_tl

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

```

1.2 Warnings

```

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

```

```

88   to the latest version of LaTeX2e.
89   }
90   \@@_msg_new:nnn {fontspec} {tu-missing}
91   {
92     The TU encoding seems to be missing; please update to the latest version of LaTeX2e.
93   }
94   \@@_msg_new:nnn {fontspec} {addfontfeatures-ignored}
95   {
96     \string\addfontfeature (s) ignored \msg_line_context;
97     it cannot be used with a font that wasn't selected by a fontspec command.\\
98     \\
99     The current font is "\use:c{font@name}".$\\
100    \int_compare:nTF { \clist_count:n {#1} = 1 }
101      { The requested feature is "#1". }
102      { The requested features are "#1". }
103    }
104    \@@_msg_new:nnn {fontspec} {feature-option-overwrite}
105    {
106      Option '#2' of font feature '#1' overwritten.
107    }
108    \@@_msg_new:nnn {fontspec} {ot-tag-too-long}
109    {
110      OpenType tag '#1' is too long; script, language, and feature tags must be four characters o
111    }
112    \@@_msg_new:nnn {fontspec} {script-not-exist}
113    {
114      Font '\l_fontspec_fontname_tl' does not contain script '#1'.
115    }
116    \@@_msg_new:nnn {fontspec} {script-not-exist-latin}
117    {
118      Font '\l_fontspec_fontname_tl' does not contain script '#1'.
119      `Script=Latin` used instead.
120    }
121    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist}
122    {
123      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
124      for AAT font '\l_fontspec_fontname_tl'.
125    }
126    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist-in-font}
127    {
128      AAT feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
129      in font '\l_fontspec_fontname_tl'.
130    }
131    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist}
132    {
133      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
134      for OpenType font '\l_fontspec_fontname_tl'
135    }
136    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist-in-font}
137    {
138      OpenType feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available

```



```

139   for font '\l_fontspec_fontname_tl'
140   with script '\l_@@_script_name_tl' and language '\l_@@_lang_name_tl'.
141   }
142   \@@_msg_new:nnn {fontspec} {no-opticals}
143   {
144     '\l_fontspec_fontname_tl' doesn't appear to have an Optical Size axis.
145   }
146   \@@_msg_new:nnn {fontspec} {language-not-exist}
147   {
148     Language '#1' not available
149     for font '\l_fontspec_fontname_tl'
150     with script '\l_@@_script_name_tl'.
151   }
152   \@@_msg_new:nnn {fontspec} {only-xetex-feature}
153   {
154     Ignored XeTeX only feature: '#1'.
155   }
156   \@@_msg_new:nnn {fontspec} {only-luatex-feature}
157   {
158     Ignored LuaTeX only feature: '#1'.
159   }
160   \@@_msg_new:nnn {fontspec} {no-mapping}
161   {
162     Input mapping not (yet?) supported in LuaTeX.
163   }
164   \@@_msg_new:nnn {fontspec} {no-mapping-ligtext}
165   {
166     Input mapping not (yet?) supported in LuaTeX.\\
167     Use "Ligatures=TeX" instead of "Mapping=tex-text".
168   }
169   \@@_msg_new:nnn {fontspec} {cm-default-obsolete}
170   {
171     The "cm-default" package option is obsolete.
172   }
173   \@@_msg_new:nnn {fontspec} {fakebold-only-xetex}
174   {
175     The "FakeBold" and "AutoFakeBold" options are only available with XeLaTeX.\\
176     Option ignored.
177   }
178   \@@_msg_new:nnn {fontspec} {font-index-needs-ttc}
179   {
180     The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
181     Feature ignored.
182   }
183   \@@_msg_new:nnn {fontspec} {feat-cannot-remove}
184   {
185     The "#1" feature cannot be deactivated. Request ignored.
186   }

```

1.3 Info messages

```

187   \@@_msg_new:nnn {fontspec} {defining-font}

```

```

188 {
189   Font family '\g_@@_nfss_family_tl' created for font '#2'
190   with options [\l_@@_all_features_clist].\\
191   \\
192   This font family consists of the following NFSS series/shapes:\\
193   \g_@@_defined_shapes_tl
194 }
195 \@@_msg_new:nnn {fontspec} {no-font-shape}
196 {
197   Could not resolve font "#1" (it probably doesn't exist).
198 }
199 \@@_msg_new:nnn {fontspec} {set-scale}
200 {
201   \l_fontspec_fontname_tl\space scale = \l_@@_scale_tl.
202 }
203 \@@_msg_new:nnn {fontspec} {setup-math}
204 {
205   Adjusting the maths setup (use [no-math] to avoid this).
206 }
207 \@@_msg_new:nnn {fontspec} {no-scripts}
208 {
209   Font "\l_fontspec_fontname_tl" does not contain any OpenType `Script' information.
210 }
211 \@@_msg_new:nnn {fontspec} {dflt-script}
212 {
213   Font "\l_fontspec_fontname_tl" falling back to default (DFLT) script.
214 }
215 \@@_msg_new:nnn {fontspec} {opa-twice}
216 {
217   Opacity set twice, in both Colour and Opacity.\\
218   Using specification "Opacity=#1".
219 }
220 \@@_msg_new:nnn {fontspec} {opa-twice-col}
221 {
222   Opacity set twice, in both Opacity and Colour.\\
223   Using an opacity specification in hex of "#1/FF".
224 }
225 \@@_msg_new:nnn {fontspec} {bad-colour}
226 {
227   Bad colour declaration "#1".
228   Colour must be one of:\\
229   * a named xcolor colour\\
230   * a six-digit hex colour RRGGBB\\
231   * an eight-digit hex colour RRGGBBTT with opacity
232 }

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

```

File V

fontspec-code-opening.dtx

1 Opening code

1.1 Package options

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

1.2 Encodings

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

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

```

39 \bool_if:NTF \g_@@_euenc_bool
40 {
41   <XE> \tl_gset:Nn \g_fontspec_encoding_tl {EU1}
42   <LU> \tl_gset:Nn \g_fontspec_encoding_tl {EU2}
43 }
44 { \tl_gset:Nn \g_fontspec_encoding_tl { TU } }
45 \tl_set:Nn \rmdefault {lmr}
46 \tl_set:Nn \sfdefault {lmss}
47 \tl_set:Nn \ttdefault {lmtt}
48 \RequirePackage[\g_fontspec_encoding_tl]{fontenc}
49 \tl_set_eq:NN \UTFencname \g_fontspec_encoding_tl % for xunicode if needed

```

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

```

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

```

Dealing with a couple of the problems introduced by babel:

```

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

```

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

```

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

```

1.3 Generic functions

```

\FontspecSetCheckBoolTrue
\FontspecSetCheckBoolFalse

```

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

```

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

```

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

```

\@@_keys_set_known:nnN

```

```

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

```

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

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

```

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

```

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

1.4 expl3 variants

```

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

```

File VI

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

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

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

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

```
\@@_primitive_font_if_null_p:N
\@@_primitive_font_if_null:NTF
13 \prg_set_conditional:Nnn \@@_primitive_font_if_null:N {p,TF,T,F}
14 {
15   \ifx #1 \nullfont
16     \prg_return_true:
17   \else
18     \prg_return_false:
19   \fi
20 }
```

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

```
\@@_primitive_font_if_exist:nTF
21 \prg_set_conditional:Nnn \@@_primitive_font_if_exist:n {TF,T,F}
22 {
23   \group_begin:
24     \@@_font_suppress_not_found_error:
25     \@@_primitive_font_set:Nnn \l_@@_primitive_font {#1} {10pt}
26     \@@_primitive_font_if_null:NTF \l_@@_primitive_font
27     { \group_end: \prg_return_false: }
28     { \group_end: \prg_return_true: }
29 }
```

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

`\@@_primitive_font_glyph_if_exist:NnTF`

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

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

`\@@_primitive_font_set_hyphenchar:Nn`

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

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

File VII

fontspec-code-interfaces.dtx

1 User commands

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

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


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

```

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

```

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

```

41 \NewDocumentCommand \newfontfamily { m O{} m O{} }
42 {
43     \@@_main_newfontfamily:nnnN {#1} {#2,#4} {#3} \NewDocumentCommand
44 }
45 \NewDocumentCommand \renewfontfamily { m O{} m O{} }
46 {
47     \@@_main_newfontfamily:nnnN {#1} {#2,#4} {#3} \RenewDocumentCommand
48 }
49 \NewDocumentCommand \setfontfamily { m O{} m O{} }
50 {
51     \@@_main_newfontfamily:nnnN {#1} {#2,#4} {#3} \DeclareDocumentCommand
52 }
53 \NewDocumentCommand \newfontface { m O{} m O{} }
54 {
55     \@@_main_newfontface:nnn {#1} {#2,#4} {#3}
56 }

```

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

```

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

```

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

```

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

```

```

76 \NewDocumentCommand \newAATfeature {mmm}
77 {
78   \@@_main_newAATfeature:nnnn {#1} {#2} {#3} {#4}
79 }
80 \NewDocumentCommand \newopentypefeature {mmm}
81 {
82   \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
83 }

```

`\newICUfeature` Deprecated.

```

84 \NewDocumentCommand \newICUfeature {mmm}
85 {
86   \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
87 }
(End definition for \newICUfeature. This function is documented on page ??.)
88 \NewDocumentCommand \aliasfontfeature {mm}
89 {
90   \@@_main_aliasfontfeature:nn {#1} {#2}
91 }
92 \NewDocumentCommand \aliasfontfeatureoption {mmm}
93 {
94   \@@_main_aliasfontfeatureoption:nnn {#1} {#2} {#3}
95 }

```

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

```

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

```

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

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

```

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

```

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

```

104 \NewDocumentCommand \DeclareFontExtensions {m}
105 {
106   \@@_main_DeclareFontExtensions:n {#1}
107 }
108 \NewDocumentCommand \IfFontFeatureActiveTF {mmm}
109 {
110   \@@_main_IfFontFeatureActiveTF:nnn {#1} {#2} {#3}
111 }

```

File VIII

fontspec-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     <XE> \fontspec_set_family:Nnn \g_@@_mathrm_tl {#1} {#2}
64     <LU> \fontspec_set_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
65     \@@_setmathrm_hook:nn {#1} {#2}
66 }

```

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

`\setboldmathrm`

```

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

```

```

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

```

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

`\setmathsf`

```

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

```

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

`\setmathtt`

```

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

```

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

Hooks:

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

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

1.2 Font feature selection

\@@_set_default_features:nn

```

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

```

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

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

```

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

```

```

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

```

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

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

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

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

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

```

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

```

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

1.3 Defining new font features

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

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

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

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

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

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

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

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

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

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

```

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

```

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

`\aliasfontfeatureoption`

```

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

```

```

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

```

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

\@@_main_DeclareFontExtensions:n

```

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

```

Defaults:

```

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

```

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

\IfFontFeatureActiveTF

```

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

```

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

File IX

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

```

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

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

```

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

```

```

67         \@@_check_ot_feat:NnnTF \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 {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_t
96
97             \@@_check_lang:NnTF \font {#1} \prg_return_true: \prg_return_false:
98         }
99         { \prg_return_false: }
100     }
101     { \prg_return_false: }
102 }

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

144     { \prg_return_false: }
145 }

```

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

```

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

```

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

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

```

146 \cs_new:Nn \fontspec_set_family:Nnn
147 {
148   \tl_set:Nn \l_@@_family_label_tl {#1}
149   \@@_select_font_family:nn {#2} {#3}
150   \tl_clear_new:N #1
151   \tl_set_eq:NN #1 \l_fontspec_family_tl
152 }
153 \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!

```

```

154 \cs_new:Nn \fontspec_set_fontface:NNnn
155 {
156   \tl_set:Nn \l_@@_family_label_tl {#1}
157   \@@_select_font_family:nn {#3}{#4}
158   \global \font #1 = \fontname \l_fontspec_font \scan_stop:
159   \tl_set_eq:NN #2 \l_fontspec_family_tl
160 }

```

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

```

\fontspec_font_if_exist:n

```

```

161 \prg_new_conditional:Nnn \fontspec_font_if_exist:n {TF,T,F}
162 {
163   \group_begin:
164     \@@_init:
165     \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
166     \@@_primitive_font_if_exist:nTF { \@@_construct_font_call:nn {#1} {} }
167     { \group_end: \prg_return_true: }
168     { \group_end: \prg_return_false: }
169 }
170 \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.

```
171 \prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
172 {
173   \exp_args:Nxx \tl_if_in:nnTF
174     { \fontname\font } { \tl_to_str:n {#1} }
175     { \prg_return_true: } { \prg_return_false: }
176 }
```

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

`\fontspec_if_small_caps:TF`

```
177 \prg_new_conditional:Nnn \fontspec_if_small_caps: {TF,T,F}
178 {
179   \@@_if_merge_shape:nTF {sc}
180   {
181     \tl_set_eq:Nc \l_@@_smcp_shape_tl { \@@_shape_merge:nn {\f@shape} {sc} }
182   }
183   {
184     \tl_set:Nn \l_@@_smcp_shape_tl {sc}
185   }
186
187   \cs_if_exist:cTF { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
188   {
189     \tl_if_eq:ccTF
190       { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
191       { \f@encoding/\f@family/\f@series/\updefault }
192       { \prg_return_false: }
193       { \prg_return_true: }
194   }
195   { \prg_return_false: }
196 }
```

(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_@@_al
22
23 <debug>\typeout{fontid: \l_@@_fontid_tl}
24
25 \@@_preparse_features:

```

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

70 \cs_new:Nn \@@_init_ttc:n
71 {
72   \str_if_eq: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 <debug>\typeout{Set~ base~ font~ for~ preliminary~ analysis: \@@_construct_font_call:nn { \l_
136 \@@_primitive_font_set:Nnn \l_fontspec_font
137 { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} } {\f@size pt}
138 \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-found} {\l_@@_fon
139 \@@_set_font_type:N \l_fontspec_font
140 <debug>\typeout{Set~ base~ font~ properly: \@@_construct_font_call:nn { \l_@@_fontname_up_tl }
141 \@@_primitive_font_gset:Nnn \l_fontspec_font
142 { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} } {\f@size pt}
143 \l_fontspec_font % this is necessary for LuaLaTeX to check the scripts properly
144 \@@_check_script:NnTF \l_fontspec_font {DFLT}
145 {
146 <debug>\typeout{Has~Opentype~scripts}
147 \bool_set_false:N \l_@@_scripts_missing_bool
148 }
149 {
150 <debug>\typeout{No~Opentype~scripts}
151 \bool_set_true:N \l_@@_scripts_missing_bool
152 }
153 }

```

(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.

```

154 \cs_new:Nn \@@_construct_font_call:nnnnnn
155 {
156 <XE> " \@@_fontname_wrap:n { #1 #2 #3 }
157 <LU> " \@@_fontname_wrap:n { #1 #2 } #3
158 #4 #5
159 \str_if_eq:eeF {#6}{ } {:#6} "
160 }

```

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

```

161 \cs_new:Nn \@@_construct_font_call:nn
162 {
163 \@@_construct_font_call:nnnnnn
164 {#1}
165 \l_@@_extension_tl
166 \l_@@_ttc_index_tl
167 \l_fontspec_renderer_tl
168 \l_@@_optical_size_tl
169 {#2}
170 }

```

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

`\@@_font_is_file:` The `\@@_fontname_wrap:n` command takes the font name and either passes it through unchanged or wraps it in the syntax for loading a font ‘by filename’. X₃TeX’s syntax is followed since luaotfload provides compatibility.

```

171 \cs_new:Nn \@@_font_is_name:
172 {
173     \cs_set_eq:NN \@@_fontname_wrap:n \use:n
174 }

175 \cs_new:Nn \@@_font_is_file:
176 {
177     \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
178 }

```

(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.

```

179 \cs_new:Nn \@@_set_scriptlang:
180 {
181     <debug> \typeout{:: _set_scriptlang:}
182     \bool_if:NT \l_@@_firsttime_bool
183     {
184         \bool_if:NTF \l_@@_scripts_missing_bool
185         {
186             <debug> \typeout{::: NADA}
187             \@@_info:n {no-scripts}
188         }
189         {
190             \tl_if_empty:NTF \l_@@_script_name_tl
191             {
192                 \@@_check_script:NnTF \l_fonts_spec_font {latn}
193                 {
194                     <debug> \typeout{::: Defaulting to latn script}
195                     \tl_set:Nn \l_@@_script_name_tl {Latin}
196                     \tl_if_empty:NT \l_@@_lang_name_tl
197                     {
198                         \tl_set:Nn \l_@@_lang_name_tl {Default}
199                     }
200                     \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
201                     \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
202                     <debug> \typeout{::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
203                 }
204             }
205             <debug> \typeout{::: Default script only}
206             \@@_info:n {dflt-script}
207             \keys_set:nx {fontspec-opentype} {Script=Default}
208             \keys_set:nx {fontspec-opentype} {Language=Default}
209         }
210     }
211     {
212         \tl_if_empty:NT \l_@@_lang_name_tl
213         {

```

```

214         \tl_set:Nn \l_@@_lang_name_tl {Default}
215     }
216 <debug> \typeout{::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
217         \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
218         \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
219     }
220 }
221 }
222 }

```

(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.

```

223 \cs_new:Nn \@@_get_features:n
224 {
225 <debug> \typeout{:: @@@_get_features:Nn { \exp_not:n {#1} } }
226 \@@_init_fontface:
227 \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#1}
228 \l_@@_keys_leftover_clist
229 \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
230 <*XE>
231 \bool_if:NTF \l_@@_ot_bool
232 {
233 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features::~"\l_@@_keys_leftover_clist}
234 \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
235 }
236 {
237 <debug> \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features::~"\l_@@_keys_leftover_c
238 \bool_if:NTF { \l_@@_atsui_bool || \l_@@_graphite_bool }
239 { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
240 }
241 </XE>
242 <*LU>
243 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features::~"\l_@@_keys_leftover_clist}
244 \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
245 </LU>
246
247 \tl_if_empty:NF \l_@@_mapping_tl
248 { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
249
250 \str_if_eq:eeF { \l_@@_hexcol_tl \l_@@_opacity_tl }
251 { \c_@@_hexcol_tl \c_@@_opacity_tl }
252 { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
253 }

```

(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.

```

254 \prg_new_conditional:Nnn \@@_save_family_needed:n { TF }
255 {
256
257 <debug> \typeout{save~ family:~ #1}
258 <debug> \typeout{== fontid_tl: "\l_@@_fontid_tl".}
259
260 \tl_if_empty:NTF \l_@@_nfss_fam_tl
261 {
262   \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
263   {
264     \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
265     \prg_return_false:
266   }
267   {
268     \tl_set:Nx \l_@@_tmp_tl {#1}
269     \tl_remove_all:Nn \l_@@_tmp_tl { ~ }
270     \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl
271     \prg_return_true:
272   }
273 }
274 {
275   \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
276   \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl _prop }
277   \prg_return_true:
278 }
279 }
280 \cs_new:Nn \@@_save_fontid_family:nn
281 {
282   \prop_get:NnNTF \g_@@_family_int_prop {#2} \l_@@_tmp_tl
283   {
284     \tl_set:Nx \l_@@_tmp_tl
285     { \int_eval:n { \l_@@_tmp_tl + 1 } }
286   }
287   { \tl_set:Nn \l_@@_tmp_tl { 0 } }
288   \prop_gput:NnV \g_@@_family_int_prop {#2} \l_@@_tmp_tl
289   \tl_gset:Nx \g_@@_nfss_family_tl { #2 ( \l_@@_tmp_tl ) }
290   \prop_gput:NnV \g_@@_fontid_family_prop {#1} \g_@@_nfss_family_tl
291 }
292 \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.

```

293 \cs_new:Nn \@@_save_family:nn
294 {
295   \@@_save_fontinfo:n {#2}
296   \@@_find_autofonts:
297   \DeclareFontFamily{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{ }

```



```

298 \@@_set_faces:
299 \@@_info:nxx {defining-font} {#1} {#2}
300 }

```

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

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

```

301 \cs_new:Nn \@@_save_fontinfo:n
302 {
303   \prop_new:c {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop}
304   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
305   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options} { \l_@@_all_feature }
306   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
307   {
308     \@@_construct_font_call:nn {\l_fontspec_fontname_tl}
309     { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
310   }
311   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
312   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num} \l_@@_language_int
313   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_fontspec_script_tag
314   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_fontspec_lang_tag
315 }

```

(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:

```

316 \cs_new:Nn \@@_find_autofonts:
317 {
318   \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
319   {
320     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
321     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
322     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspec_fontname_tl} {/BI}
323   }
324   \bool_if:NF \l_@@_nobf_bool
325   {
326     \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspec_fontname_tl} {/B}
327   }
328   \bool_if:NF \l_@@_noit_bool
329   {
330     \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_@@_fontname_bf_tl} {/I}
331   }

```

```

332     \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspec_fontname_tl} {/I}
333   }
334
335   \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
336 }

```

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

\@@_set_faces:

```

337 \cs_new:Nn \@@_set_faces:
338 {
339   \@@_add_nfssfont:nnnn \mddefault \updefault \l_fontspec_fontname_tl \l_@@_fontfeat_u
340   \@@_add_nfssfont:nnnn \bfdefault \updefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_clis
341   \@@_add_nfssfont:nnnn \mddefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_clis
342   \@@_add_nfssfont:nnnn \mddefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_clis
343   \@@_add_nfssfont:nnnn \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit_cl
344   \@@_add_nfssfont:nnnn \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl_cl
345
346   \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nnnnn ##2 }
347 }
348 \cs_new:Nn \@@_set_faces_aux:nnnnn
349 {
350   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
351   \@@_make_font_shapes:Nnnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
352 }

```

(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”).

```

353 \cs_new:Nn \fontspec_complete_fontname:Nn
354 {
355   \tl_set:Nx #1 {#2}
356   \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
357   \LU \tl_remove_all:Nn #1 {~}
358 }

```

(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

```

359 \cs_new:Nn \@@_add_nfssfont:nnnn
360 {
361   \tl_set:Nx \l_@@_this_font_tl {#3}
362
363   \tl_if_empty:xTF {#4}
364   { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
365   { \@@_keys_set_known:nxN {fontspec-preparse-nested} {#4} \l_@@_tmp_tl }

```

```

366
367 \tl_if_empty:NF \l_@@_this_font_tl
368 {
369   \prop_put:Nxx \l_@@_nfssfont_prop {#1/#2}
370   { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
371 }
372 }

```

(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_font is an AAT font or an OpenType font or a font with feature axes (either AAT or Multiple Master), respectively.

```

373 \cs_new:Nn \@@_set_font_type:N
374 {
375   <debug> \typeout{:: @@@_set_font_type:}
376   <*XE>
377   \bool_set_false:N \l_@@_tfm_bool
378   \bool_set_false:N \l_@@_atsui_bool
379   \bool_set_false:N \l_@@_ot_bool
380   \bool_set_false:N \l_@@_mm_bool
381   \bool_set_false:N \l_@@_graphite_bool
382   \ifcase\XeTeXfonttype #1
383   <debug> \typeout{::: TFM}
384   \bool_set_true:N \l_@@_tfm_bool
385   \or
386   <debug> \typeout{::: AAT}
387   \bool_set_true:N \l_@@_atsui_bool
388   \tl_if_empty:NT \l_fontspec_renderer_tl { \tl_set:Nn \l_fontspec_renderer_tl {/AAT} }
389   \ifnum\XeTeXcountvariations #1 > 0\relax
390   <debug> \typeout{::: MM}
391   \bool_set_true:N \l_@@_mm_bool
392   \fi
393   \or
394   <debug> \typeout{::: OpenType}
395   \bool_set_true:N \l_@@_ot_bool
396   \tl_if_empty:NT \l_fontspec_renderer_tl { \tl_set:Nn \l_fontspec_renderer_tl {/OT} }
397   \or
398   <debug> \typeout{::: Graphite}
399   \bool_set_true:N \l_@@_graphite_bool
400   \tl_if_empty:NT \l_fontspec_renderer_tl { \tl_set:Nn \l_fontspec_renderer_tl {/GR} }
401   \fi
402   </XE>

```

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

LuaTeX only supports one:

```

403 \*LU
404   \bool_set_true:N \l_@@_ot_bool
405 \*LU
406 }

```

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

```

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

```

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

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

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

```

407 \cs_new:Nn \@@_set_autofont:Nnn
408 {
409   \bool_if:NF \l_@@_external_bool
410   {
411     \tl_if_empty:xF {#2}
412     {
413       \tl_if_empty:NT #1
414       {
415         \@@_if_autofont:nnTF {#2} {#3}
416         { \tl_set:Nx #1 {#2#3} }
417         { \@@_info:nx {no-font-shape} {#2#3} }
418       }
419     }
420   }
421 }

```

```

422 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
423 {
424   \@@_primitive_font_set:Nnn \l_tmpa_font { \@@_construct_font_call:nn {#1} {} } {\f@size
425   \@@_primitive_font_set:Nnn \l_tmpb_font { \@@_construct_font_call:nn {#1#2} {} } {\f@size
426   \str_if_eq:eeTF { \fontname \l_tmpa_font } { \fontname \l_tmpb_font }
427   { \prg_return_false: }
428   { \prg_return_true: }
429 }

```

(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.

```

430 \cs_new:Nn \@@_make_font_shapes:Nnnnn
431 {
432   \group_begin:
433     \@@_keys_set_known:nxN {fontspec-preparse-external} { #4 } \l_@@_leftover_clist
434     \@@_load_fontname:n {#1}
435     \@@_declare_shape:nxxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
436   \group_end:
437 }
438 \cs_new:Nn \@@_load_fontname:n
439 {
440   <debug> \typeout{:: \@@_load_fontname:n {#1} }
441   \@@_load_external_fontoptions:Nn \l_fontspec_fontname_tl {#1}
442   \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
443   { \clist_clear:N \l_@@_fontopts_clist }
444   \keys_set_groups:nnV {fontspec/fontname} {getfontname} \l_@@_fontopts_clist
445   \@@_primitive_font_set:Nnn \l_fontspec_font { \@@_construct_font_call:nn { \l_fontspec_fontname_tl } }
446   \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-found} {#1} }
447 }
448 \keys_define:nn {fontspec/fontname}
449 {
450   Font .tl_set:N = \l_fontspec_fontname_tl ,
451   Font .groups:n = {getfontname} ,
452 }

```

(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>}}.

```

453 \cs_new:Nn \@@_declare_shape:nnnn
454 {
455   <debug>\typeout{~= declare_shape:~{\l_fontspec_fontname_tl}~{#1}~{#2}}
456   \tl_clear:N \l_@@_nfss_tl
457   \tl_clear:N \l_@@_nfss_sc_tl
458   \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspec_fontname_tl
459
460   \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
461
462   \@@_declare_shapes_normal:nn {#1} {#2}
463   \@@_declare_shapes_smcaps:nn {#1} {#2}
464   \@@_declare_shape_slanted:nn {#1} {#2}
465   \@@_declare_shape_loginfo:nn {#1} {#2}
466 }

```

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

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

```
\@@_setup_single_size:nn
```

```
468 \cs_new:Nn \@@_setup_single_size:nn
469 {
470   \tl_clear:N \l_@@_size_tl
471   \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
472
473   \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
474   \l_@@_sizing_leftover_clist
475   \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
476   <debug>\typeout{=== size:~\l_@@_size_tl}
477
478   % "normal"
479   \@@_load_fontname:n {\l_@@_sizedfont_tl}
480   \@@_setup_nfss:Nnnn \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
481   <debug> \typeout{=== sized~ font:~ \l_@@_sizedfont_tl}
482
483   % small caps
484   \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist
485
486   \bool_if:NF \l_@@_nosc_bool
487   {
488     \tl_if_empty:NTF \l_@@_fontname_sc_tl
489     {
490       \@@_make_smallcaps:TF
491       {
492         <debug>\typeout{====~Small~ caps~ found.}
493         \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
494       }
495       {
496         <debug>\typeout{====~Small~ caps~ not~ found.}
497         \bool_set_true:N \l_@@_nosc_bool
498       }
499     }
500     { \@@_load_fontname:n {\l_@@_fontname_sc_tl} }% local for each size
501   }
502
503   \bool_if:NF \l_@@_nosc_bool
504   {
505     \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
506     {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
507   }
508 }
```

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

```
\@@_setup_nfss:Nnnn
```

```
509 \cs_new:Nn \@@_setup_nfss:Nnnn
510 {
```

```

511 <debug>\typeout{====~Setup-NFSS~shape:~<\l_@@_size_tl>~\l_fontspeg_fontname_tl}
512
513 \@@_get_features:n { #2 , #3 , #4 }
514 <debug>\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist}
515
516 \tl_put_right:Nx #1
517 {
518   <\l_@@_size_tl> \l_@@_scale_tl
519   \@@_construct_font_call:nn { \l_fontspeg_fontname_tl }
520   { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
521 }
522 }

```

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

\@@_declare_shapes_normal:nn

```

523 \cs_new:Nn \@@_declare_shapes_normal:nn
524 {
525   \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
526   {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
527 }

```

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

\@@_declare_shapes_smcaps:nn

```

528 \cs_new:Nn \@@_declare_shapes_smcaps:nn
529 {
530   \tl_if_empty:NF \l_@@_nfss_sc_tl
531   {
532     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl} {#1}
533     { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
534   }
535 }
536 \cs_new:Nn \@@_combo_sc_shape:n
537 {
538   \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
539   { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
540   { \scdefault }
541 }

```

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

\@@_DeclareFontShape:nnnnnn

```

542 \cs_new:Nn \@@_DeclareFontShape:nnnnnn
543 {
544   <debug>\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
545   \group_begin:
546   \normalsize
547   \cs_undefine:c {#1/#2/#3/#4/\f@size}
548   \group_end:
549   \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
550   }
551 \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.

```

552 \cs_new:Nn \@@_declare_shape_slanted:nn
553 {
554   \bool_if:nT
555   {
556     \str_if_eq_p:ee {#2} {\itdefault} &&
557     !(\str_if_eq_p:ee {\itdefault} {\sldefault})
558   }
559   {
560     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}
561     {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
562   }
563 }

```

Lastly some informative messaging.

```

\@@_declare_shape_loginfo:nn 564 \cs_new:Nn \@@_declare_shape_loginfo:nn
565 {
566   \tl_gput_right:Nx \g_@@_defined_shapes_tl
567   {
568     \exp_not:n { \ }
569     -- \exp_not:N \str_case:nn {#1/#2}
570     {
571       {\mddefault/\updefault} {'normal'~}
572       {\bfdefault/\updefault} {'bold'~}
573       {\mddefault/\itdefault} {'italic'~}
574       {\mddefault/\sldefault} {'slanted'~}
575       {\bfdefault/\itdefault} {'bold~ italic'~}
576       {\bfdefault/\sldefault} {'bold~ slanted'~}
577     } (#1/#2)~
578     with~ NFSS~ spec.:~
579     \l_@@_nfss_tl
580     \exp_not:n { \ }
581     -- \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
582     {
583       {\mddefault/\scdefault} {'small~ caps'~}
584       {\bfdefault/\scdefault} {'bold~ small~ caps'~}
585       {\mddefault/\itscdefault} {'italic~ small~ caps'~}
586       {\bfdefault/\itscdefault} {'bold~ italic~ small~ caps'~}
587       {\mddefault/\slscdefault} {'slanted~ small~ caps'~}
588       {\bfdefault/\slscdefault} {'bold~ slanted~ small~ caps'~}
589     }~( #1 / \@@_combo_sc_shape:n {#2} )~
590     with~ NFSS~ spec.:~
591     \l_@@_nfss_sc_tl
592     \tl_if_empty:FF {\l_@@_postadjust_tl}
593     {
594       \exp_not:N \ and~ font~ adjustment~ code:

```



```

595         \exp_not:N \ \l_@@_postadjust_tl
596     }
597 }
598 }

```

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
599 \tl_set:Nn \l_@@_pre_feat_sclist
600 {*XE}
601 {
602     \bool_if:NT \l_@@_ot_bool
603     {
604         \tl_if_empty:NF \l_fontspec_script_tl
605         {
606             script = \l_fontspec_script_tl ;
607             language = \l_fontspec_lang_tl ;
608         }
609     }
610 }
611 {/XE}
612 {*LU}
613 {
614     mode = \l_fontspec_mode_tl ;
615     \tl_if_empty:NF \l_fontspec_script_tl
616     {
617         script = \l_fontspec_script_tl ;
618         language = \l_fontspec_lang_tl ;
619     }
620 }
621 {/LU}

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF
622 {LU}\cs_new:Nn \@@_make_smallcaps:TF
623 {XE}\cs_new:Nn \@@_make_ot_smallcaps:TF
624 {
625     \@@_check_ot_feat:NnTF \l_fontspec_font {smcp} {#1} {#2}
626 }
627 {*XE}
628 \cs_new:Nn \@@_make_smallcaps:TF
629 {
630     \bool_if:NnTF \l_@@_ot_bool
631     { \@@_make_ot_smallcaps:TF {#1} {#2} }
632     {
633         \bool_if:NT \l_@@_atsui_bool
634         { \@@_make_AAT_feature_string:NnnTF \l_fontspec_font {3}{3} {#1} {#2} }
635     }
636 }
637 {/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.

```

638 \cs_new:Nn \@@_update_featstr:n
639 {
640   <debug> \typeout{::: \@@_update_featstr:n {#1}}
641   \bool_if:NF \l_@@_firsttime_bool
642   {
643     \tl_gset:Nx \g_@@_single_feat_tl { #1 }
644     <debug> \typeout{:::~ Adding~ feature.}
645     \tl_gput_right:Nx \g_@@_rawfeatures_sclist {#1;}
646   }
647 }

```

```

\@@_remove_clashing_featstr:n 648 \cs_new:Nn \@@_remove_clashing_featstr:n
649 {
650   <debug> \typeout{::: \@@_remove_clashing_featstr:n {#1}}
651   \clist_map_inline:nn {#1}
652   {
653     <debug> \typeout{:::~ Removing~ feature~ "##1;}
654     \tl_gremove_all:Nn \g_@@_rawfeatures_sclist {##1;}
655   }
656 }

```

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: 657 \cs_set:Npn \@@_init:
658 {
659   <debug> \typeout{: \@@_init:}
660   \bool_set_false:N \l_@@_ot_bool
661   \bool_set_true:N \l_@@_firsttime_bool
662   \@@_font_is_name:
663   \tl_clear:N \l_@@_font_path_tl
664   \tl_clear:N \l_@@_optical_size_tl
665   \tl_clear:N \l_@@_ttc_index_tl
666   \tl_clear:N \l_fontspeg_renderer_tl
667   \tl_gclear:N \g_@@_defined_shapes_tl
668   \tl_gclear:N \g_@@_curr_series_tl
669   \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspeg_encoding_tl
670   <*LU>
671   \tl_set:Nn \l_fontspeg_mode_tl {node}
672   \int_set:Nn \prehyphenchar { \- } % fixme
673   \int_zero:N \posthyphenchar % fixme
674   \int_zero:N \preexhyphenchar % fixme
675   \int_zero:N \postexhyphenchar % fixme
676   </LU>
677 }

```

Executed in \@@_get_features:Nn.

```
\@@_init_fontface: 678 \cs_new:Nn \@@_init_fontface:
679 {
680   \tl_gclear:N \g_@@_rawfeatures_sclist
681   \tl_clear:N \l_@@_scale_tl
682   \tl_set_eq:NN \l_@@_opacity_tl \c_@@_opacity_tl
683   \tl_set_eq:NN \l_@@_hexcol_tl \c_@@_hexcol_tl
684   \tl_set_eq:NN \l_@@_postadjust_tl \c_@@_postadjust_tl
685   \tl_clear:N \l_@@_wordspace_adjust_tl
686   \tl_clear:N \l_@@_punctspace_adjust_tl
687 }
```

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

```
\@@_ot_validate_tag:n 688 <*LU>
689 \cs_new_protected:Nn \@@_ot_validate_tag:n
690 {
691   \@@_ot_validate_tag:w #1 \q_nil
692 }
693 \cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
694 \cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
695 {
696   \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
697     { \@@_ot_validate_tag_aux:w #2 \c_empty_tl \c_empty_tl \q_nil }
698     { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
699 }
700 \cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
701 {
702   \int_compare:nT { \tl_count:n {#5} > 2 }
703     { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
704 }
705 </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.

```
\@@_iv_str_to_num:Nn 706 <*XE>
707 \cs_new:Nn \@@_iv_str_to_num:Nn
708 {
709   \@@_strip_leading_sign:Nw #1#2 \q_nil
710 }
711 \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 s, and anything beyond four chars is snipped. The \c_empty s then are used to reconstruct the spaces in the string to number calculation.

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

```

712 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
713 {
714   \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
715     { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
716     { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
717 }

```

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.

```

718 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
719 {
720   \int_compare:nT { \tl_count:n {#6} > 2 }
721     { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }
722
723   \int_set:Nn #1
724     {
725       `#2 * "10000000
726       + `#3 * "100000
727       + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
728       + \ifx \c_empty_tl #5 32 \else `#5 \fi
729     }
730 }
731 </XE>

```

```

\@@_lang_dflt_correct:N 732 <*XE>
733 \cs_new_protected:Nn \@@_lang_dflt_correct:N
734 {
735   \int_compare:nNnT {#1} = {1145457748} % "DFLT"
736   {
737     \int_zero:N #1
738   }
739 }
740 </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: }
4 }

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

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

```

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

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

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

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

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

```

1.1 Adding features when loading fonts

When remove clashing features,

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

```

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

```

```

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

```

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

\@@_check_script:NnTF This macro takes an OpenType script tag and checks if it exists in the current font. \l_@@_script_int is used to store the number corresponding to the script tag string.

```

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

```

```

112         {
113     <debug>\typeout{:::::~ TRUE}
114         \prg_return_true:
115     }
116     {
117     <debug>\typeout{:::::~ FALSE}
118         \prg_return_false:
119     }
120 }
121 </LU>
122 }

```

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

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

```

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

```



```

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

```

(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'.

```

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

```

```

200         \int_set:Nn \l_tmpa_int {\l_tmpb_int}
201     \else
202         \int_incr:N \l_tmpa_int
203     \fi
204 }
205 \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
206 }
207 </XE>
208 <*LU>
209 {
210 <debug>\typeout{::~ fontspec_check_ot_feat:n~ {#1}}
211     \@@_ot_validate_tag:x {#2}
212     \@@_ot_validate_tag:x {#3}
213     \@@_ot_validate_tag:x {#4}
214     \cs_if_eq:NNTF #1 \font
215     { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
216     { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
217     \directlua
218     {
219         fontspec.check_ot_feat("\l_@@_tmp_tl", "#2", "#3", "#4")
220     }
221     \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
222 }
223 </LU>
224 }

```

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

1.2 OpenType feature information

```

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

```

246 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2sc}{Small~Capitals~From~Capitals}
 247 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dist}{Distances}
 248 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dlig}{Discretionary~Ligatures}
 249 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dnom}{Denominators}
 250 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dtls}{Dotless~Forms}
 251 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {expt}{Expert~Forms}
 252 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {falt}{Final~Glyph~on~Line~Alternates}
 253 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin2}{Terminal~Forms~\#2}
 254 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin3}{Terminal~Forms~\#3}
 255 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {final}{Terminal~Forms}
 256 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {flac}{Flattened~accent~forms}
 257 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {frac}{Fractions}
 258 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fwid}{Full~Widths}
 259 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {half}{Half~Forms}
 260 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {haln}{Halant~Forms}
 261 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {halt}{Alternate~Half~Widths}
 262 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hist}{Historical~Forms}
 263 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hkna}{Horizontal~Kana~Alternates}
 264 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hlig}{Historical~Ligatures}
 265 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hngl}{Hangul}
 266 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hojo}{Hojo~Kanji~Forms}
 267 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hwid}{Half~Widths}
 268 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {init}{Initial~Forms}
 269 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {isol}{Isolated~Forms}
 270 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ital}{Italics}
 271 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jalt}{Justification~Alternates}
 272 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp78}{JIS78~Forms}
 273 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp83}{JIS83~Forms}
 274 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp90}{JIS90~Forms}
 275 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp04}{JIS2004~Forms}
 276 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {kern}{Kerning}
 277 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lfbid}{Left~Bounds}
 278 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {liga}{Standard~Ligatures}
 279 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ljmo}{Leading~Jamo~Forms}
 280 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lnum}{Lining~Figures}
 281 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {loc1}{Localized~Forms}
 282 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltra}{Left-to-right~alternates}
 283 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltrm}{Left-to-right~mirrored~forms}
 284 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mark}{Mark~Positioning}
 285 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {med2}{Medial~Forms~\#2}
 286 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {medi}{Medial~Forms}
 287 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mgrk}{Mathematical~Greek}
 288 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mkmk}{Mark~to~Mark~Positioning}
 289 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mset}{Mark~Positioning~via~Substitution}
 290 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nalt}{Alternate~Annotation~Forms}
 291 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nlck}{NLC~Kanji~Forms}
 292 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nukt}{Nukta~Forms}
 293 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {numr}{Numerators}
 294 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {onum}{Oldstyle~Figures}
 295 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {opbd}{Optical~Bounds}
 296 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ordn}{Ordinals}

297 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ornm}{Ornaments}
 298 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {palt}{Proportional~Alternate~Widths}
 299 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pcap}{Petite~Capitals}
 300 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pkna}{Proportional~Kana}
 301 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pnum}{Proportional~Figures}
 302 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pref}{Pre~Base~Forms}
 303 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pres}{Pre~base~Substitutions}
 304 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstf}{Post~base~Forms}
 305 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {psts}{Post~base~Substitutions}
 306 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pwid}{Proportional~Widths}
 307 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {qwid}{Quarter~Widths}
 308 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rand}{Randomize}
 309 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rclt}{Required~Contextual~Alternates}
 310 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rkrf}{Rakar~Forms}
 311 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rlig}{Required~Ligatures}
 312 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rphf}{Reph~Forms}
 313 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtbd}{Right~Bounds}
 314 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtla}{Right-to-left~alternates}
 315 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtlm}{Right-to-left~mirrored~forms}
 316 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ruby}{Ruby~Notation~Forms}
 317 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rvrn}{Required~Variation~Alternates}
 318 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {salt}{Stylistic~Alternates}
 319 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sinf}{Scientific~Inferiors}
 320 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {size}{Optical~size}
 321 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smcp}{Small~Capitals}
 322 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smp1}{Simplified~Forms}
 323 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssNN}{Stylistic~Set~\$N\$}
 324 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssty}{Math~script~style~alternates}
 325 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {stch}{Stretching~Glyph~Decomposition}
 326 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {subs}{Subscript}
 327 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sup}{Superscript}
 328 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {swsh}{Swash}
 329 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {titl}{Titling}
 330 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing~Jamo~Forms}
 331 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional~Name~Forms}
 332 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular~Figures}
 333 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {trad}{Traditional~Forms}
 334 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third~Widths}
 335 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {unic}{Unicase}
 336 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate~Vertical~Metrics}
 337 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu~Variants}
 338 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical~Writing}
 339 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate~Vertical~Half~Metrics}
 340 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vjmo}{Vowel~Jamo~Forms}
 341 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical~Kana~Alternates}
 342 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkern}{Vertical~Kerning}
 343 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional~Alternate~Vertical~Metrics}
 344 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrt2}{Vertical~Alternates~and~Rotation}
 345 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrtr}{Vertical~Alternates~for~Rotation}
 346 \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     \@@_make_AAT_feature_string:NnnTF \l_fontspec_font {#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 57).

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-n
5   fontspec-renderer
6 }
```

Wrapper function to save some characters in the source:

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

For catching features that cannot be used in \addfontfeatures:

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

1.1 Pre-pre-parsing stages

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

Don't load font config file

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

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

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

```

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

```

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

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

```

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

```

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

```

47 \keys_define:nn {fontspec-renderer}
48 {
49 \Renderer .choices:nn =
50 {AAT,ICU,OpenType,Graphite,Full,Basic}
51 {
52 \int_compare:nTF {\l_keys_choice_int <= 4}
53 {
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-prepare} {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-prepare} {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-prepare} {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-prepare-external} {UprightFont}
104 {
105     \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
106 }

```

Italic and slanted

```

107 \@@_keys_define_code:nnn {fontspec-preparse-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-preparse-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-preparse-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-preparse-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:N { \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   \tl_set:Nx \l_@@_scale_tl { s*[ \l_@@_scale_tl ] }
274 }

```

`\@@_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.)

```

275 \cs_new:Nn \@@_calc_scale:n
276 {
277   \group_begin:
278
279   \fontencoding { \encodingdefault }
280   \fontfamily { \rmdefault }
281   \selectfont
282
283   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
284   \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_fontspec_font
285
286   \tl_set:Nx \l_@@_scale_tl
287   {
288     \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
289                 \dim_to_fp:n {\l_@@_tmpb_dim} }
290   }
291
292   \@@_info:n {set-scale}
293   \exp_args:NNNx
294   \group_end:
295   \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
296 }

```

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

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

```

297 \cs_new:Nn \@@_set_font_dimen:NnN
298 {
299   \dim_set:Nn #1 { \fontdimen #2 #3 }
300   \dim_compare:nNnT #1 = {0pt}
301   {
302     \settoheight #1
303     {
304       \str_if_eq:nnTF {#3} {\font} \rmfamily #3
305       \int_case:nnF #2
306       {
307         {5} {x} % x-height
308         {8} {X} % cap-height
309       } {?} % "else" clause; never reached.
310     }
311   }

```

```

311     }
312 }

```

(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.

```

313 \@@_keys_define_code:nnn {fontspec} {WordSpace}
314 {
315     \bool_if:NF \l_@@_firsttime_bool
316     { \_fontspec_parse_wordspace:w #1,,,\q_stop }
317 }
318 \@@_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}.

```

319 \cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
320 {
321     \tl_if_empty:nTF {#4}
322     {
323         \tl_set:Nn \l_@@_wordspace_adjust_tl
324         {
325             \fontdimen 2 \font = #1 \fontdimen 2 \font
326             \fontdimen 3 \font = #1 \fontdimen 3 \font
327             \fontdimen 4 \font = #1 \fontdimen 4 \font
328         }
329     }
330     {
331         \tl_set:Nn \l_@@_wordspace_adjust_tl
332         {
333             \fontdimen 2 \font = #1 \fontdimen 2 \font
334             \fontdimen 3 \font = #2 \fontdimen 3 \font
335             \fontdimen 4 \font = #3 \fontdimen 4 \font
336         }
337     }
338 }

```

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

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

```

339 \@@_keys_define_code:nnn {fontspec} {PunctuationSpace}
340 {
341     \str_case_e:nnF {#1}
342     {
343         {WordSpace}
344         {
345             \tl_set:Nn \l_@@_punctspace_adjust_tl
346             { \fontdimen 7 \font = 0 \fontdimen 2 \font }
347         }
348         {TwiceWordSpace}
349         {
350             \tl_set:Nn \l_@@_punctspace_adjust_tl

```

```

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

```

Secret hook into the font-adjustment code

```

360 \@@_keys_define_code:nnn {fontspec} {FontAdjustment}
361 {
362     \tl_put_right:Nx \l_@@_postadjust_tl {#1}
363 }

```

Letterspacing

```

364 \@@_keys_define_code:nnn {fontspec} {LetterSpace}
365 {
366     \@@_update_featstr:n {letterspace=#1}
367 }

```

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.

```

368 \@@_keys_define_code:nnn {fontspec} {HyphenChar}
369 {
370     \str_if_eq:nnTF {#1} {None}
371     {
372         \tl_put_right:Nn \l_@@_postadjust_tl
373         { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
374     }
375     {
376         \@@_warning:nx {only-xetex-feature} {HyphenChar}
377
378         \tl_if_single:nTF {#1}
379         { \tl_set:Nn \l_@@_hyphenchar_tl {`#1} }
380         { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }
381
382         \@@_primitive_font_glyph_if_exist:NnTF \l_fontspec_font {\l_@@_hyphenchar_tl}
383         {
384             \tl_put_right:Nn \l_@@_postadjust_tl
385             { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
386         }
387         { \@@_error:nx {no-glyph}{#1} }
388     }
389 }
390 }
391 \@@_aff_error:n {HyphenChar}

```


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

```

392 \@@_keys_define_code:nnn {fontspec} {Color}
393 {
394   \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
395   {
396     \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
397   }
398   {
399     \int_compare:nTF { \tl_count:n {#1} == 6 }
400     { \tl_set:Nn \l_@@_hexcol_tl {#1} }
401     {
402       \int_compare:nTF { \tl_count:n {#1} == 8 }
403       { \fontspec_parse_colour:viii #1 }
404       {
405         \bool_if:NF \l_@@_firsttime_bool
406         { \@@_warning:nx {bad-colour} {#1} }
407       }
408     }
409   }
410 }
411 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
412 {
413   \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
414   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
415   {
416     \bool_if:NF \l_@@_firsttime_bool
417     { \@@_warning:nx {opa-twice-col} {#7#8} }
418   }
419   \tl_set:Nn \l_@@_opacity_tl {#7#8}
420 }
421 \aliasfontfeature{Color}{Colour}
422 \@@_keys_define_code:nnn {fontspec} {Opacity}
423 {
424   \int_set:Nn \l_@@_tmp_int {255}
425   \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
426   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
427   {
428     \bool_if:NF \l_@@_firsttime_bool
429     { \@@_warning:nx {opa-twice} {#1} }
430   }
431   \tl_set:Nx \l_@@_opacity_tl
432   {
433     \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
434     \int_to_hex:n { \l_@@_tmp_int }
435   }
436 }

```

Mapping

```

437 <*XE>
438 \@@_keys_define_code:nnn {fontspec-aat} {Mapping}

```

```

439 {
440   \tl_set:Nn \l_@@_mapping_tl { #1 }
441 }
442 \@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
443 {
444   \tl_set:Nn \l_@@_mapping_tl { #1 }
445 }
446 </XE>
447 <*LU>
448 \@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
449 {
450   \str_if_eq:nnTF {#1} {tex-text}
451   {
452     \@@_warning:n {no-mapping-ligtext}
453     \msg_redirect_name:nnn {fontspec} {no-mapping-ligtext} {none}
454     \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
455   }
456   { \@@_warning:n {no-mapping} }
457 }
458 </LU>

```

1.4.1 Continuous font axes

```

459 \@@_keys_define_code:nnn {fontspec} {Weight}
460 {
461   \@@_update_featstr:n{weight=#1}
462 }
463 \@@_keys_define_code:nnn {fontspec} {Width}
464 {
465   \@@_update_featstr:n{width=#1}
466 }
467 \@@_keys_define_code:nnn {fontspec} {OpticalSize}
468 <*XE>
469 {
470   \bool_if:NTF \l_@@_ot_bool
471   {
472     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
473   }
474   {
475     \bool_if:NT \l_@@_mm_bool
476     {
477       \@@_update_featstr:n { optical size = #1 }
478     }
479   }
480   \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
481   {
482     \bool_if:NT \l_@@_firsttime_bool
483     { \@@_warning:n {no-opticals} }
484   }
485 }
486 </XE>
487 <*LU>

```

```

488 {
489   \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
490 }
491 </LU>

```

1.4.2 Font transformations

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

```

492 \keys_define:nn {fontspec}
493 {
494   FakeSlant .code:n =
495   {
496     \@@_update_featstr:n {slant=#1}
497   },
498   FakeSlant .default:n = {0.2}
499 }
500 \keys_define:nn {fontspec}
501 {
502   FakeStretch .code:n =
503   {
504     \@@_update_featstr:n {extend=#1}
505   },
506   FakeStretch .default:n = {1.2}
507 }
508 <*XE>
509 \keys_define:nn {fontspec}
510 {
511   FakeBold .code:n =
512   {
513     \@@_update_featstr:n {embolden=#1}
514   },
515   FakeBold .default:n = {1.5}
516 }
517 </XE>
518 <*LU>
519 \keys_define:nn {fontspec}
520 {
521   FakeBold .code:n = { \@@_warning:n {fakebold-only-xetex} }
522 }
523 </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.)

```

524 \keys_define:nn {fontspec}
525 {
526   AutoFakeSlant .code:n =
527   {

```

```

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

```

Same but reversed:

```

546 \keys_define:nn {fontspec}
547 {
548   AutoFakeBold .code:n =
549   {
550     \bool_if:NT \l_@@_firsttime_bool
551     {
552       \tl_set:Nn \l_@@_fake_embolden_tl {#1}
553       \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
554       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
555       \bool_set_false:N \l_@@_nobf_bool
556
557       \tl_if_empty:NF \l_@@_fake_slant_tl
558       {
559         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
560         {FakeSlant=\l_@@_fake_slant_tl}
561         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
562         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
563       }
564     }
565   },
566   AutoFakeBold .default:n = {1.5}
567 }

```

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.

```

568 \@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}
569 {
570   \@@_update_featstr:n {#1}
571 }
572 \@@_keys_define_code:nnn {fontspec-aat} {RawFeature}

```

```
573 {  
574   \@@_update_featstr:n {#1}  
575 }
```

File XIV

fontspec-code-feat-opentype.dtx

1 OpenType feature definitions

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

2 Regular key=val / tag definitions

2.1 Ligatures

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

Emulate CM extra ligatures.

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

2.2 Letters

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

```

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

```

2.3 Numbers

```

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

```

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

```

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

```

2.4 Vertical position

```

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

```

2.5 Contextuals

```

75 \@@_define_opentype_feature_group:n {Contextuals}

```

```

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

```

2.6 Diacritics

```

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

```

2.7 Kerning

```

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

```

2.8 Fractions

```

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

```



```

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

```

2.9 Style

```

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

```

2.10 CJK shape

```

149 \@@_define_opentype_feature_group:n {CJKShape}
150 \@@_define_opentype_feature:nnnnn {CJKShape} {ResetAll} {} {}
151 {
152   +trad,-trad,
153   +smpl,-smpl,
154   +jp78,-jp78,
155   +jp83,-jp83,
156   +jp90,-jp90,
157   +jp04,-jp04,
158   +expt,-expt,
159   +nlck,-nlck,
160 }
161 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Traditional} {trad} {trad} {+smpl,+jp78,+jp8}
162 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Simplified} {smpl} {smpl} {+trad,+jp78,+jp8}
163 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978} {jp78} {jp78} {+trad,+smpl,+jp8}
164 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983} {jp83} {jp83} {+trad,+smpl,+jp7}
165 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990} {jp90} {jp90} {+trad,+smpl,+jp7}

```

```

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

```

2.11 Character width

```

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

```

2.12 Vertical

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

```

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

```

3 OpenType features that need numbering

3.1 Alternate

```

196 \@@_define_opentype_feature_group:n {Alternate}
197 \keys_define:nn {fontspec-opentype}
198 {
199 Alternate .default:n = {} ,
200 Alternate / unknown .code:n =
201 {
202 \clist_map_inline:nn {#1}
203 { \@@_make_OT_feature:nnn {salt}{ +salt = ##1 }{} }

```

```

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

```

3.2 Variant / StylisticSet

```

214 \@@_define_opentype_feature_group:n {Variant}
215 \keys_define:nn {fontspec-opentype}
216 {
217     Variant .default:n = {0} ,
218     Variant / unknown .code:n =
219     {
220         \clist_map_inline:nn {#1}
221         {
222             \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
223         }
224     }
225 }
226 \aliasfontfeature{Variant}{StylisticSet}

```

3.3 CharacterVariant

```

227 \@@_define_opentype_feature_group:n {CharacterVariant}
228 \use:x
229 {
230     \cs_new:Npn \exp_not:N \fontspec_parse_cv:w
231     ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
232     {
233         \@@_make_OT_feature:xxx
234         { cv \exp_not:N \two@digits {##1} }
235         { +cv \exp_not:N \two@digits {##1} = ##2 } {}
236     }
237     \keys_define:nn {fontspec-opentype}
238     {
239         CharacterVariant / unknown .code:n =
240         {
241             \clist_map_inline:nn {##1}
242             {
243                 \exp_not:N \fontspec_parse_cv:w
244                 #####1 \c_colon_str 0 \c_colon_str \exp_not:N \q_nil
245             }
246         }
247     }
248 }

```

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

3.4 Annotation

```
249 \@@_define_opentype_feature_group:n {Annotation}
250 \keys_define:nn {fontspec-opentype}
251 {
252   Annotation .default:n = {0} ,
253   Annotation / unknown .code:n =
254   {
255     \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
256   }
257 }
```

3.5 Ornament

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

4 Script and Language

4.1 Script

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

```

291     }
292     {
293         \@@_check_script:NnTF \l_fontspec_font {latn}
294         {
295             \@@_warning:nx {script-not-exist-latn} {#1}
296             \tl_set:Nn \l_fontspec_script_tl {latn}
297             \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
298         }
299         {
300             \@@_warning:nx {script-not-exist} {#1}
301             \keys_set:nn {fontspec-opentype} { Script = Default }
302         }
303     }
304 }
305 }
306 }
307 }
308 }

```

4.2 Language

```

309 \keys_define:nn {fontspec-opentype} { Language .choice: }
310 \cs_new:Nn \fontspec_new_lang:nn
311 {
312     \keys_define:nn { fontspec-opentype } { Language / #1 .code:n =
313     {
314         \bool_set_false:N \l_@@_scriptlang_exist_bool
315         \clist_map_inline:nn {#2}
316         {
317             \@@_check_lang:NnTF \l_fontspec_font {#2}
318             {
319                 \tl_set:Nn \l_fontspec_lang_tl {#2}
320                 \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
321                 \tl_gset:Nx \g_@@_single_feat_tl { language=#2 }
322                 \bool_set_true:N \l_@@_scriptlang_exist_bool
323                 \clist_map_break:
324             }
325         }
326         \bool_if:NF \l_@@_scriptlang_exist_bool
327         {
328             \@@_warning:nx {language-not-exist} {#1}
329             \keys_set:nn {fontspec-opentype} { Language = Default }
330         }
331     }
332 }
333 }

```

Default I can't remember why this has to be special-cased. From memory, the OpenType default language is hardcoded to have a zero value, although this might be some X_YTeX-specific thing.

```

334 \@@_keys_define_code:nnn {fontspec-opentype} { Language / Default }
335 {

```

```

336 \tl_set:Nn \l_fontspec_lang_tl {DFLT}
337 \int_zero:N \l_@@_language_int
338 \tl_gset:Nn \g_@@_single_feat_tl { language=DFLT }
339 }

```

5 Backwards compatibility

```

340 \cs_new:Nn \@@_ot_compat:nn
341 {
342   \aliasfontfeatureoption {#1} {#20ff} {No#2}
343 }
344 \@@_ot_compat:nn {Ligatures} {Rare}
345 \@@_ot_compat:nn {Ligatures} {Required}
346 \@@_ot_compat:nn {Ligatures} {Common}
347 \@@_ot_compat:nn {Ligatures} {Discretionary}
348 \@@_ot_compat:nn {Ligatures} {Contextual}
349 \@@_ot_compat:nn {Ligatures} {Historic}
350 \@@_ot_compat:nn {Numbers} {SlashedZero}
351 \@@_ot_compat:nn {Contextuals} {Swash}
352 \@@_ot_compat:nn {Contextuals} {Alternate}
353 \@@_ot_compat:nn {Contextuals} {WordInitial}
354 \@@_ot_compat:nn {Contextuals} {WordFinal}
355 \@@_ot_compat:nn {Contextuals} {LineFinal}
356 \@@_ot_compat:nn {Contextuals} {Inner}
357 \@@_ot_compat:nn {Diacritics} {MarkToBase}
358 \@@_ot_compat:nn {Diacritics} {MarkToMark}
359 \@@_ot_compat:nn {Diacritics} {AboveBase}
360 \@@_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}{cyr1}
29 \newfontscript{Default}{DFLT}
30 \newfontscript{Deseret}{dsrt}
31 \newfontscript{Devanagari}{dev2,deva}
32 \newfontscript{Duployan}{dupl}
33 \newfontscript{Egyptian~Hieroglyphs}{egyp}
34 \newfontscript{Elbasan}{elba}
35 \newfontscript{Ethiopic}{ethi}
36 \newfontscript{Georgian}{geor}
37 \newfontscript{Glagolitic}{glag}
38 \newfontscript{Gothic}{goth}
39 \newfontscript{Grantha}{gran}
40 \newfontscript{Greek}{grek}
41 \newfontscript{Gujarati}{gjr2,gujr}
42 \newfontscript{Gurmukhi}{gur2,guru}
43 \newfontscript{Hangul~Jamo}{jamo}
44 \newfontscript{Hangul}{hang}
```

```

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

```



```

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

```

For convenience or backwards compatibility:

```

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

```

File XVI

fontspec-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}
```

`\newfontlanguage{Catalan}{CAT}`
`\newfontlanguage{Cebuano}{CEB}`
`\newfontlanguage{Chechen}{CHE}`
`\newfontlanguage{Chaha~Gurage}{CHG}`
`\newfontlanguage{Chattisgarhi}{CHH}`
`\newfontlanguage{Chichewa}{CHI}`
`\newfontlanguage{Chukchi}{CHK}`
`\newfontlanguage{Chipewyan}{CHP}`
`\newfontlanguage{Cherokee}{CHR}`
`\newfontlanguage{Chuvash}{CHU}`
`\newfontlanguage{Comorian}{CMR}`
`\newfontlanguage{Coptic}{COP}`
`\newfontlanguage{Cree}{CRE}`
`\newfontlanguage{Carrier}{CRR}`
`\newfontlanguage{Crimean~Tatar}{CRT}`
`\newfontlanguage{Church~Slavonic}{CSL}`
`\newfontlanguage{Czech}{CSY}`
`\newfontlanguage{Danish}{DAN}`
`\newfontlanguage{Dargwa}{DAR}`
`\newfontlanguage{Woods~Cree}{DCR}`
`\newfontlanguage{German}{DEU}`
`\newfontlanguage{Dogri}{DGR}`
`\newfontlanguage{Divehi}{DIV}`
`\newfontlanguage{Djerma}{DJR}`
`\newfontlanguage{Dangme}{DNG}`
`\newfontlanguage{Dinka}{DNK}`
`\newfontlanguage{Dungan}{DUN}`
`\newfontlanguage{Dzongkha}{DZN}`
`\newfontlanguage{Ebira}{EBI}`
`\newfontlanguage{Eastern~Cree}{ECR}`
`\newfontlanguage{Edo}{EDO}`
`\newfontlanguage{Efik}{EFI}`
`\newfontlanguage{Greek}{ELL}`
`\newfontlanguage{English}{ENG}`
`\newfontlanguage{Erzya}{ERZ}`
`\newfontlanguage{Spanish}{ESP}`
`\newfontlanguage{Estonian}{ETI}`
`\newfontlanguage{Basque}{EUQ}`
`\newfontlanguage{Evenki}{EVK}`
`\newfontlanguage{Even}{EVN}`
`\newfontlanguage{Ewe}{EWE}`
`\newfontlanguage{French~Antillean}{FAN}`
`\newfontlanguage{Farsi}{FAR}`
`\newfontlanguage{Parsi}{FAR}`
`\newfontlanguage{Persian}{FAR}`
`\newfontlanguage{Finnish}{FIN}`
`\newfontlanguage{Fijian}{FJI}`
`\newfontlanguage{Flemish}{FLE}`
`\newfontlanguage{Forest~Nenets}{FNE}`
`\newfontlanguage{Fon}{FON}`
`\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{Hawaii}{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}{KRR}
¹⁸² \newfontlanguage{Karelian}{KRL}
¹⁸³ \newfontlanguage{Karaim}{KRM}
¹⁸⁴ \newfontlanguage{Karen}{KRN}
¹⁸⁵ \newfontlanguage{Koorete}{KRT}
¹⁸⁶ \newfontlanguage{Kashmiri}{KSH}
¹⁸⁷ \newfontlanguage{Khasi}{KSI}
¹⁸⁸ \newfontlanguage{Kildin-Sami}{KSM}
¹⁸⁹ \newfontlanguage{Kui}{KUI}
¹⁹⁰ \newfontlanguage{Kulvi}{KUL}
¹⁹¹ \newfontlanguage{Kumyk}{KUM}
¹⁹² \newfontlanguage{Kurdish}{KUR}
¹⁹³ \newfontlanguage{Kurukh}{KUU}
¹⁹⁴ \newfontlanguage{Kuy}{KUY}
¹⁹⁵ \newfontlanguage{Koryak}{KYK}
¹⁹⁶ \newfontlanguage{Ladin}{LAD}
¹⁹⁷ \newfontlanguage{Lahuli}{LAH}

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

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

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


```

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

```

File XVII

fontspec-code-feat-aat.dtx

1 AAT feature definitions

These are only defined for X_YTeX.

1.1 Ligatures

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

Emulate CM extra ligatures.

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

1.2 Letters

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

1.3 Numbers

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

Both naming conventions are offered to select the number case.

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

1.4 Contextuals

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

1.5 Diacritics

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

1.6 Vertical position

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

1.7 Fractions

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

1.8 Alternate

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

```

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

```

1.9 Variant / StylisticSet

```

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

```

1.10 Style

```

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

```

1.11 CJK shape

```

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

```

```

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

```

1.12 Character width

```

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

```

1.13 Annotation

```

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

```

File XVIII

fontspec-code-enc.dtx

1 Extended font encodings

To be removed after the 2017 release of LaTeX2e:

```
1 \providecommand\UnicodeFontFile[2]{"[#1]:#2"}
2 \providecommand\UnicodeFontName[2]{"#1:#2"}
3 \langle XE \rangle \providecommand\UnicodeFontTeXLigatures[mapping=tex-text;}
4 \langle LU \rangle \providecommand\UnicodeFontTeXLigatures{+tlig;}
5 \providecommand\add@unicode@accent[2]{#2\char#1\relax}
6 \providecommand\DeclareUnicodeAccent[3]{%
7   \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}}%
8 }
```

`\EncodingCommand`

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

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

`\EncodingAccent`

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

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

`\EncodingSymbol`

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

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

`\EncodingComposite`

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

```

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

```

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

\EncodingCompositeCommand

```

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

```

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

\DeclareUnicodeEncoding

```

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

```

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

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

\UndeclareAccent 64 \DeclareDocumentCommand \UndeclareSymbol {m}

```

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

```

```

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

```

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

\UndeclareComposite

```

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

```

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

File XIX

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

```

74 \DeclareSymbolFont{operators}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\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\mddefault\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 'T_EXFont 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{gfsartemis}   {\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_{TEX}'s 'shape' font axis needs to be overloaded to support italic small caps and slanted small caps. These are the combinations to support:

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

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

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

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

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

```

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

```

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

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

```

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

```

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

1.2 Emphasis

`\emfontdeclare`

```

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

```



```

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

```

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

`\em`

```

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

```

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

`\emph`

```

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

```

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

1.3 Strong emphasis

`\strongfontdeclare`

```

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

```

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

`\strongenv`

```

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

```

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

`\strong`
`\strongreset`

```

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

```

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

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

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

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

Programmer's interface for setting nesting levels:

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

Defaults:

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

File XXII

fontspec-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\sim 20\fontspec_visible_space:%
32 }%
33 \group_end:

```

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

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

```

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

```

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

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

```

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

```

`verbatim*` With the `verbatim` package.

```

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

```

This is for vanilla \LaTeX .

```

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

```

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

```

66 \cs_set:Npn \fontspec_patch_moreverb:
67 {
68   \@ifpackageloaded{moreverb}
69   {
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 }

```

1.3 `\oldstylenums`

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

```

95 \RenewDocumentCommand \oldstylenums {m}
96 {
97   { \addfontfeature{Numbers=OldStyle} #1 }
98 }
99 \NewDocumentCommand \liningnums {m}
100 {
101   { \addfontfeature{Numbers=Lining} #1 }
102 }

```

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

```

103 </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	
\#	253, 254, 285
\,	1, 2, 3, 4, 5, 17
\-	2, 4, 124, 672
@@ commands:	
\@@_DeclareFontShape:nnnnn	525, 532, <u>542</u> , 560
\g_@@_OT_features_prop	9, 11, 72
\@@_add_nfssfont:nnnn	262, 339, 340, 341, 342, 343, 344, <u>359</u>
\@@_aff_error:n	11, 318, 359, 391
\l_@@_alias_bool	21, 204, 211, 217, 222, 229, 249
\l_@@_all_features_clist	21, 54, 99, 109, 123, 190, 305
\g_@@_all_keyval_modules_clist	1, 48, 206, 224
\g_@@_all_opentype_feature_names_prop	73, 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
\l_@@_arg_clist	60, 249, 250, 251, 254, 257
\l_@@_atsui_bool	7, 10, 238, 378, 387, 633
\l_@@_basename_tl	10, 41, 91, 356
\l_@@_bf_series_seq	46, 125, 137, 140
\g_@@_bfmathrm_tl	69, 70, 82, 87, 88, 89, 123
\@@_calc_scale:n	269, 270, <u>275</u>
\g_@@_cfg_bool	1, 7, 8, 16
\l_@@_check_bool	5, 65, 66, 194, 199, 205, 221
\l_@@_check_feat_bool	27, 55, 57, 58
\@@_check_lang:Nn	123
\@@_check_lang:NnTF	97, <u>123</u> , 317
\@@_check_lang:Nnn	127
\@@_check_lang:NnnTF	110, <u>123</u>
\@@_check_ot_feat:Nn	171
\@@_check_ot_feat:NnTF	50, 60, <u>171</u> , 625
\@@_check_ot_feat:Nnnn	176
\@@_check_ot_feat:NnnnTF	67, <u>171</u>
\@@_check_script:Nn	79
\@@_check_script:NnTF	79, 80, 144, 192, 277, 293
\@@_combo_sc_shape:n	533, 536, 581, 589
\@@_construct_font_call:nn	135, 137, 140, 142, <u>154</u> , 166, 308, 424, 425, 445, 519
\@@_construct_font_call:nnnnn	154, 163
\l_@@_curr_bfname_tl	93, 135, 145, 148, 150, 182
\l_@@_curr_fontname_tl	92, 350, 351
\g_@@_curr_series_tl	86, 124, 139, 143, 148, 150, 182, 668
\@@_declare_shape:nnnn	435, <u>453</u>
\@@_declare_shape_loginfo:nn	465, <u>564</u>
\@@_declare_shape_slanted:nn	464, <u>552</u>
\@@_declare_shapes_normal:nn	462, <u>523</u>
\@@_declare_shapes_smcaps:nn	463, <u>528</u>
\g_@@_default_fontopts_clist	47, 111, 119
\@@_define_aat_feature:nnnn	2, 3, 4, <u>5</u> , 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

<pre> \@@_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, 28, 41, 41, 42, 43, 47, 48, 60, 76, 92, 104, 110, 111, 112, 114, 119, 120, 121, 124, 147, 148, 150, 170, 193 \@@_define_opentype_feature_ group:n 1, 6, 27, 40, 59, 75, 91, 103, 113, 123, 149, 169, 187, 188, 196, 214, 227, 249, 258 \@@_define_opentype_onoffreset:nnnnn 13, 14, 15, 16, 17, 18, 33, 34, 35, 36, 37, 37, 38, 39, 50, 51, 52, 53, 54, 58, 69, 70, 71, 72, 73, 74, 85, 86, 87, 88, 89, 90, 99, 100, 101, 102, 109, 122, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 161, 162, 163, 164, 165, 166, 167, 168, 180, 181, 182, 183, 184, 185, 186, 188, 189, 190, 191, 192, 193, 194, 195 \@@_define_opentype_onreset:nnnnn 26, 45 \g_@@_defined_shapes_tl 87, 193, 566, 667 \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 37, 93, 96, 97, 100, 102, 105, 156, 159 \g_@@_em_normalise_slant_bool 25, 62, 68, 88 \g_@@_em_prop ... 60, 74, 77, 78, 94, 102 \l_@@_em_switch_tl 102, 103, 116 \l_@@_em_tmp_tl 94, 96, 117 \l_@@_emdef_int 38, 61, 77, 78, 79 \l_@@_emshape_query_tl 86, 90, 94, 97, 115 \@@_error:n 1, 10, 475 \@@_error:nn . 2, 3, 12, 14, 18, 24, 30, 36, 67, 73, 79, 85, 138, 387, 446, 703, 721 \g_@@_euenc_bool 9, 10, 18, 23, 36, 39, 58 \l_@@_ext_filename_tl 85, 86, 89, 90, 94 \l_@@_extension_tl 39, 45, 53, 72, 93, 95, 165 \l_@@_extensions_clist 48, 51, 61, 254 \l_@@_external_bool ... 22, 28, 40, 409 \@@_extract_all_features: 94 \@@_extract_all_features:n ... 20, 94 \l_@@_fake_embolden_tl 131, 535, 538, 552 </pre>	<pre> \l_@@_fake_slant_tl . 130, 530, 557, 560 \l_@@_family_fontopts_clist 53, 105, 106, 112 \g_@@_family_int_prop ... 77, 282, 288 \l_@@_family_label_tl 105, 107, 129, 148, 156 \@@_feat_off:n 37, 42 \@@_feat_prop_add:nn . 1, 2, 3, 4, 5, 5, 17 \@@_feat_reset:n 38, 43, 48 \@@_find_autofonts: 296, 316 \l_@@_firsttime_bool 1, 29, 182, 202, 271, 315, 405, 416, 428, 482, 528, 550, 641, 661 \@@_font_is_file: 30, 165, 171 \@@_font_is_name: 171, 662 \l_@@_font_path_tl ... 29, 96, 177, 663 \@@_font_suppress_not_found_ error: 5, 9, 24, 267 \l_@@_fontcfg_bool ... 11, 12, 18, 22, 81 \l_@@_fontfeat_bf_clist 63, 179, 340, 553 \l_@@_fontfeat_bfit_clist 65, 190, 343, 537, 539, 559, 561 \l_@@_fontfeat_bfsl_clist 67, 198, 344 \l_@@_fontfeat_clist 58, 130, 227, 272 \l_@@_fontfeat_curr_clist 59, 484, 493, 506 \l_@@_fontfeat_it_clist 64, 186, 341, 531 \l_@@_fontfeat_sc_clist . 68, 204, 484 \l_@@_fontfeat_sl_clist . 66, 194, 342 \l_@@_fontfeat_up_clist 62, 175, 210, 339 \g_@@_fontid_family_prop 76, 262, 290 \l_@@_fontid_tl 21, 23, 41, 97, 258, 262, 270 \l_@@_fontname_bf_tl 75, 133, 145, 321, 327, 340, 554 \l_@@_fontname_bfit_tl 76, 135, 156, 320, 321, 322, 343, 540, 562 \l_@@_fontname_bfsl_tl 137, 160, 335, 344 \l_@@_fontname_it_tl 74, 115, 134, 320, 332, 341, 532 \l_@@_fontname_sc_tl 138, 170, 488, 500 \l_@@_fontname_sl_tl 120, 136, 335, 342 \l_@@_fontname_tl .. 98, 152, 154, 158 \l_@@_fontname_up_tl 9, 41, 44, 105, 126, 132, 135, 137, 138, 140, 142 \@@_fontname_wrap:n 46, 156, 157, 173, 177 </pre>
---	---

\l_@@_fontopts_clist	\@@_load_external_fontoptions:Nn
.. 52, 102, 103, 113, 435, 442, 443, 444 18, 79, 441
\g_@@_fontopts_prop	\@@_load_font:
69, 87, 102, 105, 132, 135, 139, 140, 442	26, 132
\@@_get_features:Nn	\@@_load_fontname:n . 434, 438, 479, 500
59, 223	\@@_main_DeclareFontExtensions:n
\@@_get_features:n .. 28, 223, 273, 513 106, 252
\l_@@_graphite_bool . 10, 238, 381, 399	\@@_main_IfFontFeatureActiveTF:nnn
\c_@@_hexcol_tl 110, 257
147, 251, 683	\@@_main_addfontfeatures:n 66, 70, 144
\l_@@_hexcol_tl	\@@_main_aliasfontfeature:nn . 90, 201
..... 99, 250, 252, 396, 400, 413, 683	\@@_main_aliasfontfeatureoption:nnn
\l_@@_hyphenchar_tl 94, 220
..... 120, 379, 380, 382, 385	\@@_main_fontspec:nn
\@@_if_autofont:nn	1, 3
422	\@@_main_newAATfeature:nnnn .. 78, 175
\@@_if_autofont:nnTF	\@@_main_newfontface:nnn . 55, 112
415	\@@_main_newfontfamily:nnnN
\@@_if_detect_external:n 43, 47, 51, 99
58	\@@_main_newfontfeature:nn . 74, 168
\@@_if_detect_external:nTF 12, 58, 165	\@@_main_newopentypefeature:nnn
\@@_if_font_feature:n 82, 86, 185
263	\@@_main_setboldmathrm:nn . 27, 67
\@@_if_font_feature:nTF	\@@_main_setmainfont:nn . 7, 8, 39
261	\@@_main_setmathrm:nn
\@@_if_merge_shape:n	23, 61
25	\@@_main_setmathsf:nn
\@@_if_merge_shape:nTF	31, 73
21, 179	\@@_main_setmathtt:nn
\@@_info:n	35, 79
7, 128, 187, 206, 292	\@@_main_setmonofont:nn . 18, 43
\@@_info:nn	\@@_main_setsansfont:nn . 13, 25
8, 417	\@@_make_AAT_feature:nn . 9, 12, 76, 87
\@@_info:nnn	\@@_make_AAT_feature_string:Nnn . 26
9, 299	\@@_make_AAT_feature_string:NnnTF
\@@_init: 12, 17, 26, 634
6, 164, 268, 657	\@@_make_OT_feature:nnn
\@@_init_fontface:	32, 50, 75, 203, 210, 222, 233, 255, 264
226, 678	\@@_make_font_shapes:Nnnnn .. 351, 430
\@@_init_ttc:n	\@@_make_ot_smallcaps:TF
17, 70	622
\@@_int_mult_truncate:Nn . 74, 425	\@@_make_smallcaps:TF .. 490, 622, 628
\@@_iv_str_to_num:Nn	\l_@@_mapping_tl
..... 87, 133, 134, 183, 184, 186, 706 22, 23, 26, 143, 247, 248, 440, 444
\@@_iv_str_to_num:w . 715, 716, 718	\g_@@_math_bool
\@@_keys_define_code:nnn	5, 6,
. 7, 13, 16, 20, 24, 36, 37, 46, 81, 86,	17, 98, 100, 101, 102, 103, 104, 105,
91, 98, 103, 107, 118, 122, 127, 154,	106, 107, 110, 112, 113, 116, 118,
158, 162, 173, 177, 184, 188, 192,	119, 120, 121, 122, 123, 124, 125, 126
196, 200, 207, 212, 218, 222, 226,	\g_@@_math_euler_bool
230, 234, 238, 242, 246, 265, 313,	9, 13, 34
334, 339, 360, 364, 368, 392, 422,	\g_@@_math_lucida_bool
438, 442, 448, 459, 463, 467, 568, 572 13, 14, 14, 15, 40
\l_@@_keys_leftover_clist	\g_@@_mathrm_tl
..... 56, 124, 127, 128, 129,	63,
228, 229, 233, 234, 237, 239, 243, 244	64, 74, 75, 77, 78, 81, 84, 96, 122, 126
\@@_keys_set_known:nnN	\g_@@_mathsf_tl
.. 67, 122, 127, 129, 227, 229, 365, 433 75, 76, 79, 91, 97, 124, 127
\@@_lang_dflt_correct:N . 185, 732	\g_@@_mathtt_tl
\l_@@_lang_name_tl . 89, 140, 141, 80, 81, 82, 92, 98, 125, 128
196, 198, 201, 202, 212, 214, 216, 218	
\l_@@_language_int	
31, 45, 184, 185, 191, 197, 312, 320, 337	
\l_@@_leftover_clist . 55, 433, 435	

\l_@@_mm_bool	9, 380, 391, 475, 480	\@@_primitive_font_glyph_if_-	
\@@_msg_new:nnn	13, 18, 23, 44,	exist:Nn	30
84, 90, 94, 104, 108, 112, 116, 121,		\@@_primitive_font_glyph_if_-	
126, 131, 136, 142, 146, 152, 156,		exist:NnTF	16, 30, 382
160, 164, 169, 173, 178, 183, 187,		\@@_primitive_font_gset:Nnn . .	1, 141
195, 199, 203, 207, 211, 215, 220, 225		\@@_primitive_font_if_exist:nTF	
\@@_msg_new:nnnn 15, 28, 37, 48, 58, 66, 74		21, 166
\l_@@_never_check_bool		\@@_primitive_font_if_null:NnTF .	
.	28, 82, 129, 178, 270	13, 26, 138, 446
\g_@@_nfss_enc_tl 4, 15, 21, 33, 39, 51,		\@@_primitive_font_if_null_p:N . .	13
57, 88, 107, 240, 297, 525, 532, 560, 669		\@@_primitive_font_set:Nnn	
\l_@@_nfss_fam_tl . .	102, 244, 260, 275	1, 25, 136, 424, 425, 445
\g_@@_nfss_family_tl		\@@_primitive_font_set_hyphenchar:Nn	
. . .	41, 89, 161, 189, 264, 275, 276,	38, 373, 385
289, 290, 297, 303, 304, 305, 306,		\l_@@_proceed_bool	26, 54, 63, 67
311, 312, 313, 314, 525, 532, 560, 561		\l_@@_punctspace_adjust_tl	
\l_@@_nfss_prop	70, 148, 181	144, 149, 345, 350, 355, 686
\l_@@_nfss_sc_tl		\g_@@_rawfeatures_sclist	58,
.	100, 457, 505, 530, 533, 591	150, 276, 309, 514, 520, 645, 654, 680	
\l_@@_nfss_tl . .	101, 456, 480, 526, 579	\@@_remove_clashing_featstr:n . .	
\l_@@_nfssfont_prop	71, 346, 369	23, 69, 648
\l_@@_nobf_bool		\l_@@_rmfamily_family_tl 9, 10, 16, 152	
.	2, 26, 131, 134, 318, 325, 555	\@@_sanitise_fontname:Nn	
\l_@@_noit_bool	8, 9, 10, 44, 74, 75, 76, 84, 128
.	3, 27, 111, 114, 318, 330, 533	\@@_save_family:nn	33, 293
\l_@@_nosc_bool 4, 166, 169, 486, 497, 503		\@@_save_family_needed:n	254
\c_@@_opacity_tl 148, 251, 414, 426, 682		\@@_save_family_needed:nTF . . .	31, 254
\l_@@_opacity_tl		\@@_save_fontid_family:nn 270, 280, 292	
.	103, 250, 252, 414, 419, 426, 431, 682	\@@_save_fontinfo:n	295, 301
\l_@@_optical_size_tl		\l_@@_saved_fontname_tl 106, 458, 471	
.	104, 168, 472, 489, 664	\l_@@_scale_tl	
\l_@@_options_tl . . .	105, 151, 154, 158	107, 201, 272, 273, 286, 295, 518, 681
\l_@@_ot_bool	8, 28, 39,	\l_@@_script_int . . .	30, 42, 94, 134,
65, 78, 91, 108, 121, 136, 231, 269,		136, 142, 186, 190, 197, 280, 297, 311	
379, 395, 404, 470, 480, 602, 630, 660		\l_@@_script_name_tl	84, 139,
\@@_ot_compat:nn . . .	340, 344, 345,	140, 150, 190, 195, 200, 202, 216, 217	
346, 347, 348, 349, 350, 351, 352,		\l_@@_scriptlang_exist_bool	
353, 354, 355, 356, 357, 358, 359, 360		24, 274, 281, 286, 314, 322, 326
\@@_ot_validate_tag:n		\l_@@_scripts_missing_bool	
.	105, 158, 159, 211, 212, 213, 688	29, 147, 151, 178, 184, 272
\@@_ot_validate_tag:w	691, 694	\@@_select_font_family:nn	
\@@_ot_validate_tag_aux:w 697, 698, 700		1, 43, 149, 154, 157, 157
\g_@@_pkg_euler_loaded_bool 2, 3, 8, 15		\@@_set_autofont:Nnn	
\c_@@_postadjust_tl	149, 684	320, 321, 322, 327, 332, 335, 407
\l_@@_postadjust_tl	146, 362,	\@@_set_default_features:nn . .	60, 116
372, 384, 526, 533, 561, 592, 595, 684		\@@_set_faces:	298, 337
\l_@@_pre_feat_sclist 151, 309, 520, 599		\@@_set_faces_aux:nnnnn	346, 348
\@@_preparse_features:	25, 118	\@@_set_font_default_features:nnn	
\l_@@_prev_unicode_name_tl . . .	57, 62	61, 121
\l_@@_primitive_font	25, 26	\@@_set_font_dimen:NnN . .	283, 284, 297

\@@_set_font_type:N	9, 27, 38, 64, 77, 90, 107, 120, 135, 139, 373
\@@_set_scriptlang:	27, 179
\@@_setboldmathrm_hook:nn	71, 91
\@@_setmainfont_hook:nn	22, 85
\@@_setmathrm_hook:nn	65, 88
\@@_setmathsf_hook:nn	77, 89
\@@_setmathtt_hook:nn	83, 90
\@@_setmonofont_hook:nn	58, 87
\@@_setsansfont_hook:nn	40, 86
\@@_setup_nfss:Nnnn	480, 505, 509
\@@_setup_single_size:nn	460, 468
\l_@@_sffamily_family_tl	27, 28, 34, 153
\@@_shape_merge:nn	10, 11, 12, 13, 14, 15, 16, 17, 18, 22, 28, 33, 181, 538, 539
\g_@@_single_feat_tl	60, 76, 90, 265, 277, 279, 281, 282, 321, 338, 643
\l_@@_size_tl	108, 232, 470, 475, 476, 511, 518
\l_@@_sizedfont_tl	109, 236, 471, 479, 481
\l_@@_sizefeat_clist	49, 50, 209, 214, 364, 370
\l_@@_sizing_leftover_clist	57, 474, 480, 506
\l_@@_smcp_shape_tl	121, 181, 184, 187, 190
\@@_strip_leading_sign:Nw	709, 712
\@@_strip_plus_minus:n	194, 196
\@@_strip_plus_minus_aux:Nq	196, 197
\l_@@_strnum_int	32, 87, 93, 133, 144, 183, 198, 280, 297, 320
\l_@@_strong_int	39, 134, 137, 138, 141, 143, 146, 157, 160
\g_@@_strong_prop	75, 116, 124, 125, 135, 143
\l_@@_strong_switch_tl	119, 143, 144
\l_@@_strong_tmp_tl	118, 135, 137
\l_@@_strongdef_int	40, 117, 124, 125, 126
\@@_swap_plus_minus:n	70, 76
\@@_swap_plus_minus_aux:Nq	76, 77
\l_@@_tfm_bool	6, 377, 384
\l_@@_this_feat_clist	61, 250, 258, 263
\l_@@_this_font_tl	110, 215, 216, 220, 248, 257, 263, 361, 367, 370
\l_@@_tmp_int	33, 424, 425, 433, 434
\l_@@_tmp_tl	41, 42, 44, 45, 93, 94, 107, 108, 109, 110, 111, 123, 124, 127, 128, 132, 135, 138, 139, 139, 140, 150, 161, 162, 165, 215, 216, 219, 262, 264, 268, 269, 270, 282, 284, 285, 287, 288, 289, 365
\l_@@_tmpa_bool	19, 63, 66, 68
\l_@@_tmpa_dim	43, 283, 288
\l_@@_tmpa_fp	41
\l_@@_tmpa_int	34, 137, 139, 142, 147, 150
\l_@@_tmpa_tl	28, 29, 53, 112
\l_@@_tmpb_dim	44, 284, 289
\l_@@_tmpb_fp	42
\l_@@_tmpb_int	35, 135, 139, 147
\l_@@_tmpb_tl	34, 39, 42, 46, 50, 53, 113, 132, 133, 134, 135
\l_@@_tmpc_dim	45
\l_@@_tmpc_int	36, 141, 144
\@@_trace:n	10
\l_@@_ttc_index_tl	95, 96, 100, 101, 114, 166, 665
\l_@@_ttfamily_family_tl	45, 46, 52, 154
\@@_update_featstr:n	19, 72, 172, 248, 252, 366, 461, 465, 477, 496, 504, 513, 570, 574, 638
\@@_warning:n	3, 4, 15, 29, 35, 94, 452, 456, 483, 521
\@@_warning:nn	5, 22, 62, 63, 68, 164, 218, 250, 290, 295, 300, 328, 376, 406, 417, 429
\@@_warning:nnn	6, 34, 181, 191
\l_@@_wordspace_adjust_tl	145, 149, 323, 331, 685
\	17, 25, 33, 33, 34, 37, 38, 39, 97, 98, 99, 166, 175, 180, 190, 191, 192, 217, 222, 228, 229, 230, 568, 580, 594, 595
\	34
A	
\acute	18
\addfontfeature	31, 47, 68, 96, 97, 101
\addfontfeatures	64, 76, 144
\advance	74
\aliasfontfeature	35, 88, 90, 201, 213, 226, 421
\aliasfontfeatureoption	55, 56, 57, 92, 220, 342
\AtBeginDocument	43, 53, 113, 125, 132
\author	36
B	
\bar	22

<code>\bfdefault</code>	49, 74, 78, 81, 84, 88, 91, 92, 139, 140, 143, 340, 343, 344, 572, 575, 576, 584, 586, 588
<code>\bfseries</code>	161
<code>\bgroup</code>	37
<code>\boldmath</code>	28
bool commands:	
<code>\bool_gset_false:N</code>	3, 6, 8, 10, 68, 98, 100, 101, 102, 103, 104, 105, 106, 107, 112, 113, 118, 119, 120, 121, 122, 123, 124, 125
<code>\bool_gset_true:N</code>	2, 5, 7, 9, 9, 13, 14, 15, 36, 62
<code>\bool_if:NTF</code>	1, 8, 10, 11, 17, 23, 23, 28, 29, 34, 35, 39, 39, 40, 40, 58, 58, 65, 66, 67, 68, 72, 78, 78, 81, 82, 84, 88, 91, 97, 100, 108, 111, 121, 126, 129, 136, 153, 167, 182, 184, 202, 205, 217, 221, 231, 249, 272, 286, 315, 325, 326, 330, 405, 409, 416, 428, 470, 475, 482, 486, 503, 528, 550, 602, 630, 633, 641
<code>\bool_if:nTF</code>	178, 238, 318, 480, 554, 696, 714
<code>\bool_lazy_and:nnTF</code>	27
<code>\bool_new:N</code>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
<code>\bool_set_false:N</code>	18, 22, 29, 57, 61, 63, 63, 66, 90, 110, 114, 116, 134, 138, 147, 169, 194, 204, 222, 271, 274, 314, 377, 378, 379, 380, 381, 533, 555, 660
<code>\bool_set_true:N</code>	12, 26, 27, 28, 54, 55, 59, 65, 66, 94, 111, 131, 146, 151, 153, 166, 199, 211, 229, 269, 270, 281, 322, 384, 387, 391, 395, 399, 404, 497, 661
<code>\bool_until_do:nn</code>	91, 139, 195
<code>\breve</code>	23
C	
<code>\char</code>	5, 6, 17
char commands:	
<code>\char_set_catcode_active:n</code>	28, 30
<code>\char_set_catcode_ignore:n</code>	233
<code>\char_set_catcode_space:n</code>	17
<code>\check</code>	24
clist commands:	
<code>\clist_clear:N</code>	103, 106, 272, 443
<code>\clist_count:N</code>	251
<code>\clist_count:n</code>	100
<code>\clist_gput_right:Nn</code>	118
<code>\clist_gset:Nn</code>	1, 118
<code>\clist_map_break:</code>	54, 66, 283, 323
<code>\clist_map_inline:Nn</code>	48, 61, 206, 224
<code>\clist_map_inline:nn</code>	74, 74, 86, 121, 124, 202, 220, 241, 275, 315, 460, 651
<code>\clist_new:N</code>	47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68
<code>\clist_put_left:Nn</code>	493
<code>\clist_put_right:Nn</code>	210, 531, 537, 539, 553, 559, 561
<code>\clist_set:Nn</code>	50, 99, 109, 175, 179, 186, 190, 194, 198, 204, 209, 214, 249, 254, 364
<code>\clist_set_eq:NN</code>	250, 484
<code>\colon</code>	30, 31, 114
<code>\convertcolorspec</code>	396
cs commands:	
<code>\cs:w</code>	127
<code>\cs_end:</code>	127
<code>\cs_generate_variant:Nn</code>	11, 12, 73, 75, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 153, 292, 467, 551, 693, 711
<code>\cs_gset:Npn</code>	29
<code>\cs_if_eq:NNTF</code>	106, 160, 214
<code>\cs_if_exist:NTF</code>	3, 27, 187, 394
<code>\cs_if_exist_p:N</code>	30
<code>\cs_new:Nn</code>	1, 1, 1, 4, 5, 5, 7, 7, 10, 11, 12, 13, 14, 15, 15, 19, 20, 25, 37, 38, 38, 39, 43, 44, 45, 50, 61, 67, 67, 70, 73, 74, 76, 79, 79, 94, 94, 99, 112, 116, 118, 121, 132, 144, 146, 154, 154, 159, 160, 161, 168, 171, 175, 175, 179, 185, 196, 201, 220, 223, 252, 257, 268, 275, 280, 293, 297, 301, 310, 316, 337, 340, 348, 353, 359, 373, 407, 430, 438, 453, 468, 509, 523, 528, 536, 542, 552, 564, 622, 623, 628, 638, 648, 678, 707
<code>\cs_new:Npn</code>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 65, 66, 77, 197, 230
<code>\cs_new_protected:Nn</code>	1, 689, 733
<code>\cs_new_protected:Npn</code>	58, 114
<code>\cs_set:Npn</code>	1, 5, 9, 50, 54, 60, 61, 66, 70, 72, 81, 88, 111, 112, 113, 152, 153, 177, 319, 411, 657, 694, 700, 712, 718
<code>\cs_set_eq:NN</code>	31, 43, 60, 63, 85, 85, 86, 87, 88, 89, 90, 91, 92, 170, 173

\cs_to_str:N	101, 106, 108, 127, 162, 216
\cs_undefine:N	86, 276, 547
\cyrillicencoding	51, 55, 83
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, 297
\DeclareFontShape	46, 48, 50, 52, 53, 53, 54, 549
\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
\DeclareUnicodeEncoding	21, 39
\def	13, 28, 34, 42
\defaultfontfeatures	57
\Delta	53
dim commands:	
\dim_compare:nNnTF	300
\dim_new:N	43, 44, 45
\dim_set:Nn	299
\dim_to_fp:n	288, 289
dim internal commands:	
__dim_eval:w	76
__dim_eval_end:	76
\directlua	6, 110, 163, 217
\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, 96, 201, 727, 728
else commands:	
\else:	34
\em	83, 110
\emfontdeclare	58, 162
\emminnershape	110, 162
\emph	110
\emreset	74, 106, 110
\emshape	110, 162
\EncodingAccent	15
\EncodingCommand	9
\EncodingComposite	27
\EncodingCompositeCommand	33
\encodingdefault	21, 39, 57, 78, 279
\EncodingSymbol	21
\endinput	8, 13, 22, 29
environments:	
listingcont*	66
verbatim*	50
\ExecuteOptions	21
exp commands:	
\exp_after:wN	473
\exp_args:NNNx	293
\exp_args:Nnnx	193
\exp_args:Nnx	41, 42, 43, 47, 48
\exp_args:No	89
\exp_args:NV	281
\exp_args:Nx	64, 69, 460
\exp_args:Nxx	173
\exp_not:N	15, 16, 17, 20, 33, 34, 35, 51, 52, 53, 83, 104, 106, 107, 108, 230, 231, 234, 235, 243, 244, 276, 277, 569, 581, 594, 595
\exp_not:n	13, 31, 49, 52, 60, 225, 260, 568, 580
\expandafter	29
F	
\familydefault	20, 38, 56
\FancyVerbSpace	85
\fi	10, 19, 29, 32, 36, 98, 98, 110, 116, 203, 392, 401, 727, 728
fi commands:	
\fi:	36
file commands:	
\file_if_exist:nTF	25, 89
\file_input:n	90

\filedate	56	\fontspec_if_current_script:n ..	116
\fileversion	56	\fontspec_if_current_script:nTF	116
\fmtname	29	\fontspec_if_feature:n	34
\font	3, 6, 7, 9, 9, 12, 16, 27, 38, 41, 50, 64, 67, 77, 80, 90, 97, 106, 107, 110, 120, 135, 158, 160, 174, 214, 283, 304, 325, 326, 327, 333, 334, 335, 346, 351, 356, 373, 385	\fontspec_if_feature:nnn	60
font commands:		\fontspec_if_feature:nnnTF	60
\l_fontspec_font 17, 41, 60, 136, 138, 139, 141, 143, 144, 158, 192, 277, 284, 293, 317, 382, 445, 446, 625, 634		\fontspec_if_feature:nTF	34
\l_tmpa_font	424, 426	\fontspec_if_fontspec_font:	1
\l_tmpb_font	425, 426	\fontspec_if_fontspec_font:TF .	1,
\fontdimen	79, 79, 299, 325, 326, 327, 333, 334, 335, 346, 351, 356	7, 25, 36, 62, 75, 88, 105, 118, 133, 147	
\fontdimen8	78	\fontspec_if_language:n	86
\fontencoding	4, 15, 33, 51, 107, 279	\fontspec_if_language:nn	103
\fontfamily	16, 29, 34, 52, 106, 161, 280	\fontspec_if_language:nnTF	103
\fontname	39, 158, 174, 426	\fontspec_if_language:nTF	86
\fontshape	7, 22, 23	\fontspec_if_opentype:	23
\fontspec	1, 25, 31, 41, 125	\fontspec_if_opentype:TF	23
fontspec commands:		\fontspec_if_script:n	73
\fontspec_calc_scale:n	77	\fontspec_if_script:nTF	73
\fontspec_complete_fontname:Nn .		\fontspec_if_small_caps:	177
.....	105, 115,	\fontspec_if_small_caps:TF	177
120, 135, 156, 160, 170, 236, 350, 353		\l_fontspec_lang_tl	
\g_fontspec_default_fontopts_tl .	31	.. 48, 142, 173, 314, 319, 336, 607, 618	
\g_fontspec_encoding_tl		\fontspec_maybe_setup_maths:	94
.....	23, 41, 42, 44, 48, 49, 51, 52, 55, 56, 74, 75, 77, 78, 79, 79, 80, 81, 84, 87, 88, 89, 91, 92, 669	\fontspec_merge_shape:n	
\l_fontspec_family_tl	19, 41, 46, 51, 56
.....	41, 41, 78, 151, 159	\l_fontspec_mode_tl	
\l_fontspec_feature_string_tl	19, 53	71, 75, 76, 85, 614, 671
\fontspec_font_if_exist:n	161	\fontspec_new_lang:nn	102, 310
\fontspec_font_if_exist:nTF	170	\fontspec_new_script:nn	98, 268
\l_fontspec_fontname_tl		\fontspec_parse_colour:niii .	403, 411
.....	8, 18, 21, 41, 44, 46, 81, 102, 114, 118, 124, 126, 129, 134, 139, 144, 149, 201, 209, 213, 308, 322, 327, 332, 339, 441, 442, 445, 450, 455, 458, 511, 519, 532, 540, 554, 562	\fontspec_parse_cv:w	230, 243
\fontspec_if_aat_feature:nn	5	_fontspec_parse_wordspace:w	316, 319
\fontspec_if_aat_feature:nnTF	5	\fontspec_patch_fancyvrb: ...	47, 81
\fontspec_if_current_feature:n .	171	\fontspec_patch_listings: ...	48, 88
\fontspec_if_current_feature:nTF		\fontspec_patch_moreverb: ...	46, 66
.....	171, 281	\fontspec_patch_verbatim: ...	45, 50
\fontspec_if_current_language:n	131	\fontspec_print_visible_spaces:	
\fontspec_if_current_language:nTF		27, 42, 56, 62, 77
.....	131	\l_fontspec_renderer_tl	51,
		55, 59, 60, 80, 167, 388, 396, 400, 666	
		\l_fontspec_script_tl	
		47, 95, 125, 140,
		173, 279, 296, 313, 604, 606, 615, 617	
		\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	154
		\fontspec_set_strong_level:n ...	160
		\fontspec_setup_maths:	1, 129
		\fontspec_tmp:	60, 63

\fontspec_visible_space: 14, 31, 85, 92	\int_if_even:nTF 37
\fontspec_visible_space_fallback: 18, 20	\int_incr:N 79, 97, 100, 126, 141, 150, 202
fontspec internal commands:	\int_new:N 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40
\l_fontspec_check_bool 90, 94, 100, 111, 138, 146, 153, 167	\int_set:Nn 11, 42, 45, 76, 78, 88, 94, 95, 96, 135, 137, 141, 147, 159, 160, 187, 200, 280, 297, 320, 424, 672, 723
__fontspec_update_featstr:n 97	\int_to_hex:n 434
\FONTSPECCTX 2	\int_use:N 78, 93, 97, 102, 125, 134, 138, 143
\FontspecSetCheckBoolFalse 65	\int_zero:N 61, 89, 105, 117, 137, 146, 156, 157, 193, 337, 673, 674, 675, 737
\FontspecSetCheckBoolTrue 65	\l_tmpa_int 89, 91, 93, 95, 97, 193, 195, 198, 200, 202
fp commands:	\l_tmpb_int ... 88, 91, 95, 187, 195, 200
\fp_eval:n 288	\itdefault 2, 11, 13, 15, 41, 77, 84, 89, 341, 343, 556, 557, 561, 573, 575
\fp_new:N 41, 42	\itscdefault 2, 7, 11, 13, 15, 16, 17, 585, 586
	\itshape 38, 66, 112
G	K
\g 115	keys commands:
\Gamma 52	\l_keys_choice_int 52, 57, 73
\gdef 2	\keys_define:nn 3, 3, 7, 9, 20, 20, 22, 27, 47, 69, 81, 92, 170, 197, 207, 212, 215, 230, 237, 237, 244, 250, 259, 267, 270, 309, 312, 448, 492, 500, 509, 519, 524, 546
\GetFileInfo 55	\keys_if_choice_exist:nnnTF . 180, 190
\global 7, 74, 158	\keys_if_exist:nnTF 177, 187, 208, 226, 234, 241
\grave 19	\l_keys_key_tl 123, 128, 133, 138
group commands:	\keys_set:nn 9, 13, 32, 42, 79, 83, 88, 200, 201, 207, 208, 213, 217, 218, 231, 234, 238, 239, 244, 245, 301, 329, 454
\group_begin: 4, 23, 27, 28, 56, 72, 119, 149, 163, 266, 277, 432, 545	\keys_set_groups:nnn 444
\group_end: 27, 28, 33, 33, 39, 81, 128, 160, 167, 168, 274, 294, 436, 548	\keys_set_known:nn 15
	\keys_set_known:nnN .. 70, 80, 150, 473
H	\l_keys_value_tl ... 123, 128, 133, 138
\hat 25, 113	L
\hbar 71	\l 31, 51, 53, 63, 64, 65, 70
\hbox 36	\Lambda 55
\hyphenchar 6, 9	\latinencoding 52, 56, 84
	\leavevmode 36
I	\let 38
\IfBooleanTF 118, 130	\liningnums 95
\ifcase 382	listingcont*(environment) 66
\IfFontExistsTF 170	\LuaLaTeX 34
\IfFontFeatureActiveTF 108, 257	
\ifmmode 36	
\IfNoValueTF 59	
\ifnum 6, 93, 197, 389	
\ifx 15, 29, 30, 98, 110, 116, 727, 728	
\ignorespaces 4, 9, 14, 19, 62, 166	
\InputIfFileExists 3	
int commands:	
\int_case:nn 57, 73	
\int_case:nnTF 305	
\int_compare:nNnTF 144, 735	
\int_compare:nTF 32, 52, 100, 251, 399, 402, 433, 702, 720	
\int_compare_p:nNn 91, 139, 195	
\int_eval:n 285	

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, 115
<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, 115
<code>\mathsf</code>	79, 91
<code>\mathtt</code>	80, 92
<code>\mddefault</code>	74, 75, 77, 79, 80, 87, 89, 339, 341, 342, 571, 573, 574, 583, 585, 587
<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>	453
<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, 99, 100, 104, 108
<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	
\newICUfeature	84
\newopentypefeature	80, 185
\normalfont	23, 41, 59, 73, 120, 155
\normalsize	50, 546
\null	36
\nullfont	15
\numexpr	44
O	
\oldstylenums	4, 95, 126
\Omega	62
\or	385, 393, 397
P	
\par	75
Path	24
\Phi	60
\Pi	57
\postexhyphenchar	675
\posthyphenchar	673
\preexhyphenchar	674
\prehyphenchar	672
prg commands:	
\prg_new_conditional:Nnn	
.	1, 5, 23, 25, 26, 30, 34, 58, 60, 73, 79, 86, 103, 116, 123, 127, 131, 161, 171, 171, 176, 177, 254, 263, 422
\prg_return_false:	
.	3, 13, 16, 18, 20, 27, 28, 30, 31, 35, 36, 50, 51, 53, 57, 67, 68, 69, 71, 80, 82, 84, 97, 99, 100, 101, 110, 112, 114, 118, 125, 125, 127, 129, 140, 142, 144, 153, 167, 168, 174, 175, 192, 195, 205, 221, 265, 279, 282, 427
\prg_return_true:	3, 13, 16, 28, 28, 33, 36, 50, 54, 67, 68, 80, 83, 97, 100, 110, 114, 125, 125, 130, 140, 153, 167, 167, 174, 175, 179, 193, 205, 221, 271, 277, 282, 428
\prg_set_conditional:Nnn	13, 21
\ProcessOptions	22
prop commands:	
\prop_gc clear:N	60, 116
\prop_get:NnN	
.	41, 44, 47, 48, 93, 95, 123, 138, 151, 152
\prop_get:NnNTF	85, 86, 94, 102, 102, 105, 132, 135, 143, 262, 282, 442
\prop_gput:Nnn	11, 78, 84, 125, 135, 140, 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, 288, 289, 290, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 304, 305, 305, 306, 306, 307, 308, 309, 310, 311, 311, 312, 312, 313, 313, 314, 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
\prop_gput_if_new:Nnn	77, 83, 124
\prop_gremove:Nn	139
\prop_if_in:NnTF	9, 87
\prop_map_inline:Nn	346
\prop_new:N	
.	69, 70, 71, 72, 73, 74, 75, 76, 77, 303
\prop_put:Nnn	81, 82, 148, 181, 369
\providecommand	1, 2, 2, 3, 3, 4, 5, 6
\ProvidesExplFile	49
\ProvidesExplPackage	44, 45, 46
\Psi	61
Q	
\quad	56
quark commands:	
\q_nil	
.	76, 77, 196, 197, 231, 244, 691, 694, 697, 698, 700, 709, 712, 715, 716, 718
\q_stop	316, 319
R	
\relax	5, 6, 29, 36, 44, 45, 50, 55, 74, 389

`\renewcommand` 71
`\RenewDocumentCommand` 47, 95
`\renewfontfamily` 45
`\RequirePackage` 5, 43, 48, 48, 62
`\RequirePackageWithOptions` 7, 12
`\rmdefault` . 10, 20, 29, 45, 96, 115, 126, 280
`\rmfamily` 13, 78, 304

S

scan commands:

`\scan_stop:` 3, 7, 17, 32, 40, 158
`\scdefault` 2, 3, 11, 12,
 13, 14, 17, 18, 51, 538, 539, 540, 583, 584
`\scshape` 38
`\select` 20
`\selectfont`
 .. 5, 7, 17, 22, 23, 35, 53, 108, 161, 281

seq commands:

`\seq_if_empty:NTF` 137
`\seq_new:N` 46
`\seq_put_right:Nn` 125, 140
`\setboldmathrm` 25, 28, 67, 93, 115
`\setfontfamily` 49
`\setmainfont` 6, 7, 25, 28, 113
`\SetMathAlphabet`
 ... 77, 78, 79, 80, 84, 87, 88, 89, 91, 92
`\setmathrm` 21, 61, 92, 115
`\setmathsf` 29, 73, 94
`\setmathtt` 33, 79, 95
`\setmonofont` 16, 43
`\setromanfont` 37
`\setsansfont` 11, 25
`\SetSymbolFont` 17, 75, 81
`\settoheight` 302
`\sfdefault` 28, 38, 46, 97, 127
`\sffamily` 31
`\Sigma` 58
`\sishape` 2
`\sldefault` 3, 12,
 14, 16, 46, 342, 344, 557, 560, 574, 576
`\slscdefault` .. 3, 12, 14, 15, 16, 18, 587, 588
`\slshape` 38, 64
`\space` 34, 39, 44, 201

str commands:

`\c_backslash_str` 87
`\c_colon_str` 231, 244
`\str_case:nn` 78, 569, 581
`\str_case:nnTF` 199, 267
`\str_case_e:nnTF` 341

`\str_if_eq:eeTF`
 .. 20, 38, 56, 72, 93, 159, 250, 288, 426
`\str_if_eq:nnTF`
 87, 124, 139, 304, 370, 450
`\str_if_eq_p:ee` 556, 557
`\str_if_eq_p:nn` 696, 714
`\str_lower_case:n` 72, 93
`\string` 21, 56, 76, 96
`\strong` 151
`\strongenv` 130, 151
`\strongfontdeclare` 114, 161
`\strongreset` 121, 147, 151
`\suppressfontnotfounderror` 11
sys commands:
`\sys_if_engine luatex:TF` 3
`\sys_if_engine xetex:TF` 10

T

TeX and L^AT_EX 2_ε commands:

`\@` 31, 59
`\@sverb` 40, 42
`\@filelist` 50
`\@ifpackageloaded`
 1, 6, 13, 14, 15, 52, 68,
 83, 90, 96, 100, 101, 102, 103, 104,
 105, 106, 107, 108, 112, 113, 114,
 118, 119, 120, 121, 122, 123, 124, 125
`\@ifstar` 40
`\@makeother` 38
`\@noligs` 39
`\@nomath` 85, 132
`\@onlypreamble` 92, 93, 94, 95
`\@sverb` 42
`\@sxverbatim` 62
`\@tempa` 29, 30
`\@verb` 40
`\@verbatim` 56, 62, 77
`\add@unicode@accent` 5, 7, 19
`\color@` 394
`\curr@fontshape` 107, 161, 215
`\define@ant@mathversions` 98
`\define@iwona@mathversions` 110
`\define@kurier@mathversions` 116
`\f@encoding` 32, 187, 190, 191
`\f@family` ... 3, 3, 32, 41, 44, 47, 48,
 93, 95, 123, 138, 151, 152, 187, 190, 191
`\f@series`
 ... 23, 32, 124, 135, 138, 187, 190, 191
`\f@shape` 22, 23, 28, 33, 77, 86, 181
`\f@size` 107,
 137, 142, 161, 215, 424, 425, 445, 547

<code>\listing@line</code>	74	<code>\tl_if_in:NnTF</code>	50, 50, 254
<code>\lst@visiblespace</code>	92	<code>\tl_if_in:nnTF</code>	64, 65, 66, 173
<code>\not@math@alphabet</code> ...	6, 40, 45, 50, 55	<code>\tl_if_single:nTF</code>	126, 378
<code>\reset@font</code>	153	<code>\tl_new:N</code> .	78, 79, 80, 81, 85, 86, 87,
<code>\thelisting@line</code>	74		88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
<code>\two@digits</code>	222, 234, 235		98, 99, 100, 101, 102, 103, 104, 105,
<code>\verb@eol@error</code>	38		106, 107, 108, 109, 110, 111, 112,
<code>\verbatim@font</code>	39		113, 114, 115, 116, 117, 118, 119,
<code>\verbatim@line</code>	75		120, 121, 122, 123, 124, 125, 129,
<code>\verbatim@processline</code>	72		130, 131, 132, 133, 134, 135, 136,
<code>\verbatim@start</code>	56, 77		137, 138, 139, 140, 141, 142, 143,
<code>\xll@defaultthyphenchar</code>	7, 13		144, 145, 146, 150, 151, 152, 153, 154
<code>\z@</code>	6	<code>\tl_put_left:Nn</code>	46
tex commands:		<code>\tl_put_right:Nn</code>	134, 362, 372, 384, 516
<code>\tex_hyphenchar:D</code>	40	<code>\tl_remove_all:Nn</code> ...	47, 86, 269, 357
<code>\tex_iffontchar:D</code>	32	<code>\tl_remove_once:Nn</code>	52
<code>\textsc</code>	37	<code>\tl_replace_all:Nnn</code>	90, 92, 356
<code>\textsf</code>	33	<code>\tl_set:Nn</code>	21,
<code>\textsi</code>	2		22, 26, 28, 29, 34, 39, 39, 42, 45, 46,
<code>\textvisiblespace</code>	24		46, 47, 53, 53, 55, 58, 71, 84, 85, 86,
<code>\the</code>	75		89, 95, 96, 100, 101, 107, 108, 127,
<code>\Theta</code>	54		148, 156, 161, 162, 184, 195, 198,
<code>\tilde</code>	21		214, 215, 216, 216, 220, 232, 244,
<code>\title</code>	32		268, 272, 273, 279, 284, 286, 287,
tl commands:			295, 296, 319, 323, 331, 336, 345,
<code>\c_empty_tl</code>			350, 355, 355, 361, 379, 380, 388,
.....	60, 697, 698, 715, 716, 727, 728		396, 400, 400, 413, 416, 419, 431,
<code>\tl_clear:N</code>			440, 444, 472, 489, 530, 552, 599, 671
.....	23, 45, 107, 133, 248, 258, 456, 457,	<code>\tl_set_eq:NN</code>	10, 21, 28,
.....	470, 663, 664, 665, 666, 681, 685, 686		39, 41, 46, 49, 51, 52, 55, 56, 57, 57,
<code>\tl_clear_new:N</code>	82, 83, 84, 150		62, 126, 145, 151, 159, 181, 257, 458,
<code>\tl_const:Nn</code>	11,		471, 532, 540, 554, 562, 682, 683, 684
.....	12, 13, 14, 15, 16, 17, 18, 147, 148, 149	<code>\tl_to_str:N</code>	21
<code>\tl_count:n</code>	399, 402, 702, 720	<code>\tl_to_str:n</code>	87, 174
<code>\tl_gclear:N</code>	265, 667, 668, 680	<code>\tl_trim_spaces:n</code>	14, 16
<code>\tl_gput_right:Nn</code>	566, 645	<code>\tl_use:N</code>	22, 33, 539
<code>\tl_gremove_all:Nn</code>	654	<code>\tmpa</code>	28, 29
<code>\tl_gset:Nn</code>	41, 42,	token commands:	
.....	44, 60, 76, 96, 97, 98, 124, 126, 127,	<code>\token_to_str:N</code>	87, 394
.....	128, 139, 240, 282, 289, 321, 338, 643	<code>\ttdefault</code>	46, 47, 56, 98, 128
<code>\tl_gset_eq:NN</code>	264, 275, 669	<code>\ttfamily</code>	49
<code>\tl_if_empty:NnTF</code>		<code>\typeout</code>	3,
..	29, 46, 50, 82, 190, 196, 212, 215,		5, 23, 31, 37, 52, 59, 60, 69, 71, 75,
.....	247, 260, 279, 367, 388, 396, 400,		81, 83, 86, 93, 96, 97, 109, 113, 117,
.....	413, 475, 488, 530, 535, 557, 604, 615		120, 123, 134, 134, 135, 138, 140,
<code>\tl_if_empty:nTF</code>	7, 14, 18, 57, 88, 89,		146, 146, 150, 150, 154, 181, 182,
.....	90, 109, 129, 138, 164, 321, 363, 411, 592		186, 194, 202, 203, 205, 210, 210,
<code>\tl_if_eq:NnTF</code>	189, 414, 426		216, 225, 228, 233, 236, 237, 243,
<code>\tl_if_eq:nnTF</code>	91, 143		243, 253, 256, 257, 258, 259, 260,
<code>\tl_if_exist:NnTF</code>	538		276, 277, 375, 383, 386, 390, 394,
<code>\tl_if_exist_p:N</code>	28		

