The fontspec package Font selection for X¬ETEX and LuaETEX

WILL ROBERTSON and KHALED HOSNY will.robertson@latex-project.org

2017/01/02 v2.5c

Contents

Ι	Getting started	6
1	History	6
2	Introduction 2.1 Acknowledgements	6
3	Package loading and options 3.1 Font encodings	7 7 7 7 8
4	Interaction with LaTeX 2ε and other packages 4.1 Verbatim 4.2 Discretionary hyphenation: \- 4.3 Commands for old-style and lining numbers 4.4 Italic small caps 4.5 Emphasis and nested emphasis	8 8 8 8 8
II	General font selection	10
5	Font selection 5.1 By font name	10 II II I2
6	Commands to select font families 6.1 More control over font shape selection	

	6.3 6.4	Choosing additional NFSS font faces	16 17
7	Miscel	laneous font selecting details	18
III	Sele	ecting font features	20
8	Defaul	lt settings	20
9	Defaul	It settings from a file	21
10	Worki	ng with the currently selected features Priority of feature selection	22 22
11	Differe	ent features for different font shapes	23
12	Selecti	ing fonts from TrueType Collections (TTC files)	24
13	Differe	ent features for different font sizes	25
14	Font in 14.1 14.2 14.3 14.4 14.5 14.6 14.7 14.8	Colour Scale Interword space Post-punctuation space The hyphenation character Optical font sizes Font transformations Letter spacing	26 26 27 28 28 28 30 30 31
IV	Ope	enType	32
15	Introd	uction	32
	15.1 15.2	How to select font features	32 33
16	Open	Type font features	34
	16.1	Tag-based features	34
	16.2	Letters	34
	16.3	Style	41
	16.4	Diacritics	43
	16.5	Kerning	43
	16.6	Character width	43
	16.7 16.8	Vertical typesetting	46 46
	16.0	Numeric features	46 48
	10.U	A DELIVOR SUIDIS AND TANGUAGES	/1 ()

V	Commands for accents and symbols ('encodings')	53
17	A new Unicode-based encoding from scratch	53
18	Adjusting a pre-existing encoding	54
19	Summary of commands	56
V]	LuaTEX-only font features	57
20	Custom font features	57
V]	I Fonts and features with X _H T _E X	59
21	X¬TEX-only font features	59
	21.1 Mapping	
	21.2 Different font technologies: AAT and OpenType	59
	21.3 Optical font sizes	59
22	Mac OS X's AAT fonts	6o
	22.1 Ligatures	60
	22.2 Letters	
	22.3 Numbers	60
	22.4 Contextuals	61
	22.5 Vertical position	61
	22.6 Fractions	61
	22.7 Variants	61
	22.8 Alternates	63
	22.9 Style	63
	22.10 CJK shape	63
	22.11 Character width	64
	22.12 Vertical typesetting	64
	22.13 Diacritics	64
	22.14 Annotation	64
V	II Customisation and programming interface	65
23	Defining new features	65
24	Defining new scripts and languages	66
25	Going behind fontspec's back	66
26	Renaming existing features & options	66
27	Programming interface	67

	27.2 27.3		67 68
IX	Imp	plementation	70
28	Loadir	ng	70
29	Declar	ration of variables and functions	70
	29.1	Generic functions	72
	29.2	The state of the s	73
30	Error/	warning/info messages	73
	30.1	Errors	74
	30.2	TAT	75
	30.3	Info messages	77
31	Openi	ng code	78
	31.1	· ·	78
	31.2	Encodings	78
32	expl3	interface for font loading	8o
33	User c	ommands	81
34	Progra	ammer's interface	88
35	Intern	als	93
	35.1	The main function for setting fonts	93
	35.2	Setting font shapes in a family	00
	35.3	Initialisation	09
	35.4	Miscellaneous	09
36	Open 7	Type definitions code	10
	36.1	Adding features when loading fonts	ΙI
	36.2	OpenType feature information	14
37	Graph	ite/AAT code	17
38	Font lo	pading (keyval) definitions	18
	38.1	OpenType feature definitions	32
	38.2	Regular key=val / tag definitions	32
	38.3	OpenType features that need numbering	
	38.4	Script and Language	38
	38.5	Backwards compatibility	40
	38.6	Font script definitions	
	38.7	Font language definitions	42
	38.8	AAT feature definitions	49
39	Extend	ded font encodings	53

40	Selecti	ing maths fonts	155
41	Closin		159
	4I.I	Compatibility	159
	41.2	Finishing up	159
42	Patchi	ng code	159
	42.I	Italic small caps and so on	159
	42.2	Emphasis	160
	42.3	\	161
	42.4	Verbatims	162
	42.5	\oldstylenums	164

Part I

Getting started

1 History

This package began life as a LaTeX interface to select system-installed Mac OS X fonts in Jonathan Kew's XaTeX, the first widely-used Unicode extension to TeX. Over time, XaTeX was extended to support OpenType fonts and then was ported into a cross-platform program to run also on Windows and Linux.

More recently, LuaTeX is fast becoming the TeX engine of the day; it supports Unicode encodings and OpenType fonts and opens up the internals of TeX via the Lua programming language. Hans Hagen's ConTeXt Mk. IV is a re-write of his powerful typesetting system, taking full advantage of LuaTeX's features including font support; a kernel of his work in this area has been extracted to be useful for other TeX macro systems as well, and this has enabled fontspec to be adapted for LaTeX when run with the LuaTeX engine.

2 Introduction

The fontspec package allows users of either X₄T_EX or LuaT_EX to load OpenType fonts in a Late X document. No font installation is necessary, and font features can be selected and used as desired throughout the document.

Without fontspec, it is necessary to write cumbersome font definition files for LTEX, since LTEX's font selection scheme (known as the 'NFSS') has a lot going on behind the scenes to allow easy commands like \emph or \bfseries. With an uncountable number of fonts now available for use, however, it becomes less desirable to have to write these font definition (.fd) files for every font one wishes to use.

Because fontspec is designed to work in a variety of modes, this user documentation is split into separate sections that are designed to be relatively independent. Nonetheless, the basic functionality all behaves in the same way, so previous users of fontspec under X₃T_EX should have little or no difficulty switching over to LuaT_EX.

This manual can get rather in-depth, as there are a lot of details to cover. See the documents fontspec-example.tex for a complete minimal example to get started quickly.

2.1 Acknowledgements

This package could not have been possible without the early and continued support the author of X₃T₂X, Jonathan Kew. When I started this package, he steered me many times in the right direction.

I've had great feedback over the years on feature requests, documentation queries, bug reports, font suggestions, and so on from lots of people all around the world. Many thanks to you all.

Thanks to David Perry and Markus Böhning for numerous documentation improvements and David Perry again for contributing the text for one of the sections of this manual.

Special thanks to Khaled Hosny, who was the driving force behind the support for LuaLTEX, ultimately leading to version 2.0 of the package.

3 Package loading and options

For basic use, no package options are required:

\usepackage{fontspec}

Package options will be introduced below; some preliminary details are discussed first.

3.1 Font encodings

The 2016 release of fontspec initiated some changes for font encodings and the loading of xunicode. The 2017 release rolls out those changes as default.

The now-default tuenc package option switches the NFSS font encoding to TU. TU is a new Unicode font encoding, intended for both XaTeX and LuaTeX engines, and automatically contains support for symbols covered by LaTeX's traditional T1 and TS1 font encodings (for example, \%, \textbullet, \"u, and so on). As a result, with this package option, Ross Moore's xunicode package is **not** loaded. Some new, experimental, features are now provided to customise some encoding details; see Part V on page 53 for further details.

Pre-2017 behaviour can be achieved with the euenc package option. This selects the EU1 or EU2 encoding (X\(\frac{1}{2}\)EX/LuaT\(\frac{1}{2}\)X, resp.) and loads the xunicode package. Package authors and users who have referred explicitly to the encoding names EU1 or EU2 should update their code or documents. (See internal variable names described in Section 27 on page 67 for how to do this properly.)

3.2 Maths fonts adjustments

By default, fontspec adjusts LEX's default maths setup in order to maintain the correct Computer Modern symbols when the roman font changes. However, it will attempt to avoid doing this if another maths font package is loaded (such as mathpazo or the unicode-math package).

If you find that fontspec is incorrectly changing the maths font when it shouldn't be, apply the no-math package option to manually suppress its behaviour here.

3.3 Configuration

If you wish to customise any part of the fontspec interface, this should be done by creating your own fontspec.cfg file, which will be automatically loaded if it is found by X=TEX or LuaTEX. A fontspec.cfg file is distributed with fontspec with a small number of defaults set up within it.

To customise fontspec to your liking, use the standard .cfg file as a starting point or write your own from scratch, then either place it in the same folder as the main document for isolated cases, or in a location that $X_{\exists}T_{\exists}X$ or LuaTeX searches by default; e.g. in MacTeX: ~/Library/texmf/tex/latex/.

The package option ${\tt no-config}$ will suppress the loading of the fontspec.cfg file under all circumstances.

3.4 Warnings

This package can give some warnings that can be harmless if you know what you're doing. Use the quiet package option to write these warnings to the transcript (.log) file instead.

Use the silent package option to completely suppress these warnings if you don't even want the .log file cluttered up.

4 Interaction with \LaTeX 2 ε and other packages

This section documents some areas of adjustment that fontspec makes to improve default behaviour with \LaTeX 2ε and third-party packages.

4.1 Verbatim

Many verbatim mechanisms assume the existence of a 'visible space' character that exists in the ASCII space slot of the typewriter font. This character is known in Unicode as U+2423: BOX OPEN, which looks like this: '__'.

When a Unicode typewriter font is used, LTEX no longer prints visible spaces for the verbatim* environment and \verb* command. This problem is fixed by using the correct Unicode glyph, and the following packages are patched to do the same: listings, fancyvrb, moreverb, and verbatim.

In the case that the typewriter font does not contain $'_{\square}$, the Latin Modern Mono font is used as a fallback.

4.2 Discretionary hyphenation: \-

\- ETEX defines the macro \- to insert discretionary hyphenation points. However, it is hard-coded in ETEX to use the hyphen - character. Since fontspec provides features to change the hyphenation character on a per font basis, the definition of \- is changed to adapt accordingly.

4.3 Commands for old-style and lining numbers

\oldstylenums \liningnums

Large for the end of the end of

4.4 Italic small caps

\itshape \slshape \scshape

Note that this package redefines the \itshape, \slshape, and \scshape commands in order to allow them to select italic small caps in conjunction. With these changes, writing \itshape\scshape will lead to italic small caps, and \upshape subsequently

then moves back to small caps only. \upshape again returns from small caps to upright regular. (And similarly for for \slshape. In addition, once italic small caps are selected then \slshape will switch to slanted small caps, and vice versa.)

4.5 Emphasis and nested emphasis

\eminnershape

 $\text{ETEX } 2_{\mathcal{E}}$ allows you to specify the behaviour of \emph nested within \emph by setting the \eminnershape command. For example,

\renewcommand\eminnershape{\upshape\scshape}

will produce small caps within $\epsilon \in \{1, ...\}$.

\emfontdeclare

The fontspec package takes this idea one step further to allow arbitrary font changes (e.g., boldness) and arbitrary levels of nesting within emphasis. This is performed using the \emfontdeclare command, which takes a comma-separated list of font switches corresponding to increasing levels of emphasis. Two examples:

- I. \emfontdeclare{\itshape, \upshape\scshape, \itshape} will lead to 'italics', 'small caps', then 'italic small caps' as the level of emphasis increases, as long as italic small caps are defined for the font. Note that \upshape is required because the font changes are cascading.
- 2. \emfontdeclare{\bfseries,\fontseries{h}\selectfont,\fontseries{x}\selectfont} could lead to (if fonts are set up correctly) 'bold', 'heavy', and 'extra bold'.

The implementation of these feature tries to be 'smart' and guess what level of emphasis to use in the case of manual font changing. This is reliable only if you use series-and/or shape- changing commands in \emfontdeclare. For example:

```
\emfontdeclare{\itshape,\upshape\scshape,\itshape}
...
\scshape small caps \emph{hello}
```

Here, the emphasised text 'hello' will be printed in italic small caps since \emph can detect that the current font shape is already in the second 'mode' of emphasis.

\emreset

Finally, if you have so much nested emphasis that \emfontdeclare runs out of options, it will insert \emreset (by default just \upshape) and start again from the beginning.

Part II

General font selection

This section concerns the variety of commands that can be used to select fonts.

These are the main font-selecting commands of this package. The \fontspec command selects a font for one-time use; all others should be used to define the standard fonts used in a document, as shown in Example I. Here, the scales of the fonts have been chosen to equalise their lowercase letter heights. The Scale font feature will be discussed further in Section 14 on page 26, including methods for automatic scaling.

Note that while these commands all look and behave largely identically, the default setup for font loading automatically adds the Ligatures=TeX feature for the \set-mainfont and \setsansfont commands. These defaults (and further customisations possible) are discussed in Section 8 on page 20.

The font features argument accepts comma separated $\langle font\ feature \rangle = \langle option \rangle$ lists; these are described later:

- For general font features, see Section 14 on page 26
- For OpenType fonts, see Part IV on page 32
- For X₇T_FX-only general font features, see Part VII on page 59
- For LuaTEX-only general font features, see Part VI on page 57
- For features for AAT fonts in X¬TEX, see Section 22 on page 60

5 Font selection

In both LuaTEX and XETEX, fonts can be selected either by 'font name' or by 'file name', but there are some differences in how each engine finds and selects fonts — don't be too surprised if a font invocation in one engine needs correction to work in the other.

Example 1: Loading the default, sans serif, and monospaced fonts.

```
\setmainfont{texgyrebonum-regular.otf}
\setsansfont{lmsans10-regular.otf}[Scale=MatchLowercase]
\setmonofont{Inconsolata.otf}[Scale=MatchLowercase]

\rmfamily Pack my box with five dozen liquor jugs\par
\sffamily Pack my box with five dozen liquor jugs\par
\ttfamily Pack my box with five dozen liquor jugs
```

Pack my box with five dozen liquor jugs Pack my box with five dozen liquor jugs Pack my box with five dozen liquor jugs

5.1 By font name

Fonts known to LuaTeX or XaTeX may be loaded by their standard names as you'd speak them out loud, such as Times New Roman or Adobe Garamond. 'Known to' in this case generally means 'exists in a standard fonts location' such as ~/Library/Fonts on Mac OS X, or C:\Windows\Fonts on Windows. In LuaTeX, fonts found in the Texms tree can also be loaded by name.

The simplest example might be something like

```
\setmainfont{Cambria}[ ... ]
```

in which the bold and italic fonts will be found automatically (if they exist) and are immediately accessible with the usual \textit and \textbf commands.

The 'font name' can be found in various ways, such as by looking in the name listed in a application like Font Book on Mac OS X. Alternatively, TeXLive contains the otfinfo command line program, which can query this information; for example:

```
otfinfo -a `kpsewhich lmroman1o-regular.otf` results in 'LM Roman 10'.
```

LuaTEX users only In order to load fonts by their name rather than by their filename (*e.g.*, 'Latin Modern Roman' instead of 'ec-lmrro'), you may need to run the script luaotfload-tool, which is distributed with the luaotfload package. Note that if you do not execute this script beforehand, the first time you attempt to typeset the process will pause for (up to) several minutes. (But only the first time.) Please see the luaotfload documentation for more information.

5.2 By file name

X\(\text{TEX}\) and LuaT\(\text{EX}\) also allow fonts to be loaded by file name instead of font name. When you have a very large collection of fonts, you will sometimes not wish to have them all installed in your system's font directories. In this case, it is more convenient to load them from a different location on your disk. This technique is also necessary in X\(\text{TEX}\) when loading OpenType fonts that are present within your T\(\text{EX}\) distribution, such as \(/\text{usr}/\)local/texlive/2013/texmf-dist/fonts/opentype/public. Fonts in such locations are visible to X\(\text{TEX}\) but cannot be loaded by font name, only file name; LuaT\(\text{EX}\) does not have this restriction.

When selecting fonts by file name, any font that can be found in the default search paths may be used directly (including in the current directory) without having to explicitly define the location of the font file on disk.

Fonts selected by filename must include bold and italic variants explicitly.

```
\setmainfont{texgyrepagella-regular.otf}[
    BoldFont = texgyrepagella-bold.otf ,
    ItalicFont = texgyrepagella-italic.otf ,
    BoldItalicFont = texgyrepagella-bolditalic.otf ]
```

fontspec knows that the font is to be selected by file name by the presence of the '.otf' extension. An alternative is to specify the extension separately, as shown following:

```
\setmainfont{texgyrepagella-regular}[
    Extension = .otf ,
    BoldFont = texgyrepagella-bold ,
    ...]
```

If desired, an abbreviation can be applied to the font names based on the mandatory 'font name' argument:

```
\setmainfont{texgyrepagella}[
    Extension = .otf ,
    UprightFont = *-regular ,
    BoldFont = *-bold ,
    ... ]
```

In this case 'texgyrepagella' is no longer the name of an actual font, but is used to construct the font names for each shape; the * is replaced by 'texgyrepagella'. Note in this case that UprightFont is required for constructing the font name of the normal font to use.

To load a font that is not in one of the default search paths, its location in the filesystem must be specified with the Path feature:

Note that X-TEX and LuaTEX are able to load the font without giving an extension, but fontspec must know to search for the file; this can can be indicated by using the Path feature without an argument:

```
\setmainfont{texgyrepagella-regular}[
    Path, BoldFont = texgyrepagella-bold,
    ...]
```

My preference is to always be explicit and include the extension; this also allows fontspec to automatically identify that the font should be loaded by filename.

In previous versions of the package, the Path feature was also provided under the alias ExternalLocation, but this latter name is now deprecated and should not be used for new documents.

5.3 Querying whether a font 'exists'

```
\label{lem:linear_linear_linear_linear} $$ \prod_{f \in \mathcal{F}(f)} {\langle font \ name \rangle} {\langle fune \ branch \rangle} {\langle false \ branch \rangle} $$
```

The conditional \IfFontExistsTF is provided to test whether the $\langle font \ name \rangle$ exists or is loadable. If it is, the $\langle true \ branch \rangle$ code is executed; otherwise, the $\langle false \ branch \rangle$ code is.

This command can be slow since the engine may resort to scanning the filesystem for a missing font. Nonetheless, it has been a popular request for users who wish to define 'fallback fonts' for their documents for greater portability.

In this command, the syntax for the \(\)font name\(\) is a restricted/simplified version of the font loading syntax used for \fontspec and so on. Fonts to be loaded by filename are detected by the presence of an appropriate extension (.otf, etc.), and paths should be included inline. E.g.:

```
\IffontExistsTF{cmr10}{T}{F}
\IffontExistsTF{Times New Roman}{T}{F}
\IffontExistsTF{texgyrepagella-regular.otf}{T}{F}
\IffontExistsTF{/Users/will/Library/Fonts/CODE2000.TTF}{T}{F}
```

The \IfFontExistsTF command is a synonym for the programming interface function \fontspec_font_if_exist:nTF (Section 27 on page 67).

6 Commands to select font families

For cases when a specific font with a specific feature set is going to be re-used many times in a document, it is inefficient to keep calling \fontspec for every use. While the \fontspec command does not define a new font instance after the first call, the feature options must still be parsed and processed.

\newfontfamily

For this reason, new commands can be created for loading a particular font family with the \newfontfamily command, demonstrated in Example 2. This macro should be used to create commands that would be used in the same way as \rmfamily, for example. If you would like to create a command that only changes the font inside its argument (i.e., the same behaviour as \emph) define it using regular Lagar commands:

```
\newcommand\textnote[1]{{\notefont #1}}
\textnote{This is a note.}
```

Note that the double braces are intentional; the inner pair are used to to delimit the scope of the font change.

\newfontface

Sometimes only a specific font face is desired, without accompanying italic or bold variants being automatically selected. This is common when selecting a fancy italic font, say, that has swash features unavailable in the upright forms. \newfontface is used for this purpose, shown in Example 3, which is repeated in Section 22.4 on page 61.

Comment for advanced users: The commands defined by \newfontface and \newfontfamily include their encoding information, so even if the document is set to use a legacy TFX encoding, such commands will still work correctly. For example,

```
Example 2: Defining new font families.

\text{newfontfamily\notefont{Kurier}}

This is a note.
\text{notefont This is a \emph{note}.}
```

Example 3: Defining a single font face.

\newfontface\fancy{Hoefler Text Italic}%
 [Contextuals={WordInitial,WordFinal}]
\fancy where is all the vegemite
% \emph, \textbf, etc., all don't work

where is all the vegemite

```
\documentclass{article}
\usepackage{fontspec}
\newfontfamily\unicodefont{Lucida Grande}
\usepackage{mathpazo}
\usepackage[T1]{fontenc}
\begin{document}
A legacy \TeX\ font. {\unicodefont A unicode font.}
\end{document}
```

6.1 More control over font shape selection

```
BoldFont = \langle font name \rangle
ItalicFont = \langle font name \rangle
BoldItalicFont = \langle font name \rangle
SlantedFont = \langle font name \rangle
BoldSlantedFont = \langle font name \rangle
SmallCapsFont = \langle font name \rangle
```

The automatic bold, italic, and bold italic font selections will not be adequate for the needs of every font: while some fonts mayn't even have bold or italic shapes, in which case a skilled (or lucky) designer may be able to chose well-matching accompanying shapes from a different font altogether, others can have a range of bold and italic fonts to chose among. The BoldFont and ItalicFont features are provided for these situations. If only one of these is used, the bold italic font is requested as the default from the new font. See Example 4.

If a bold italic shape is not defined, or you want to specify both custom bold and italic shapes, the <code>BoldItalicFont</code> feature is provided.

Example 4: Explicit selection of the bold font.

| The Neuron Contraction | The Neuron Contract

6.1.1 Small caps and slanted font shapes

When a font family has both slanted and italic shapes, these may be specified separately using the analogous features SlantedFont and BoldSlantedFont. Without these, however, the Lagarantee for slanted (\textsl, \slshape) will default to the italic shape.

Pre-OpenType, it was common for font families to be distributed with small caps glyphs in separate fonts, due to the limitations on the number of glyphs allowed in the PostScript Type I format. Such fonts may be used by declaring the SmallCapsFont of the family you are specifying:

```
\setmainfont{Minion MM Roman}[
   SmallCapsFont={Minion MM Small Caps & Oldstyle Figures}]
Roman 123 \\ \textsc{Small caps 456}
```

In fact, you should specify the small caps font for each individual bold and italic shape as in

```
\setmainfont{ <upright> }[
   UprightFeatures = { SmallCapsFont={ <sc> } } ,
   BoldFeatures = { SmallCapsFont={ <bf sc> } } ,
   ItalicFeatures = { SmallCapsFont={ <it sc> } } ,
   BoldItalicFeatures = { SmallCapsFont={ <bf it sc> } } ,
   BoldItalicFeatures = { SmallCapsFont={ <bf it sc> } } ,
}
```

For most modern fonts that have small caps as a font feature, this level of control isn't generally necessary.

All of the bold, italic, and small caps fonts can be loaded with different font features from the main font. See Section II for details. When an OpenType font is selected for SmallCapsFont, the small caps font feature is not automatically enabled. In this case, users should write instead, if necessary,

```
\setmainfont{\...}[
   SmallCapsFont={\...},
   SmallCapsFeatures={Letters=SmallCaps},
]
```

6.2 Specifically choosing the NFSS family

In LTEX's NFSS, font families are defined with names such as 'ppl' (Palatino), 'lmr' (Latin Modern Roman), and so on, which are selected with the \fontfamily command:

```
\fontfamily{ppl}\selectfont
```

In fontspec, the family names are auto-generated based on the fontname of the font; for example, writing \fontspec{Times New Roman} for the first time would generate an internal font family name of 'TimesNewRoman(1)'. Please note that should not rely on the name that is generated.

In certain cases it is desirable to be able to choose this internal font family name so it can be re-used elsewhere for interacting with other packages that use the LaTeX's font selection interface; an example might be

```
\usepackage{fancyvrb}
\fvset{fontfamily=myverbatimfont}
```

To select a font for use in this way in fontspec use the NFSSFamily feature:

\newfontfamily\verbatimfont[NFSSFamily=myverbatimfont]{Inconsolata}

It is then possible to write commands such as:

```
\fontfamily{myverbatimfont}\selectfont
```

which is essentially the same as writing \verbatimfont, or to go back to the orginal example:

```
\fvset{fontfamily=myverbatimfont}
```

Only use this feature when necessary; the in-built font switching commands that fontspec generates (such as \verbatimfont in the example above) are recommended in all other cases.

If you don't wish to explicitly set the NFSS family but you would like to know what it is, an alternative mechanism for package writers is introduced as part of the fontspec programming interface; see the function \fontspec_set_family:Nnn for details (Section 27 on page 67).

6.3 Choosing additional NFSS font faces

LTEX's font selection scheme (NFSS) is more flexible than the fontspec interface discussed up until this point. It assigns to each font face a family (discussed above), a series such as bold or light or condensed, and a shape such as italic or slanted or small caps. The fontspec features such as BoldFont and so on all assign faces for the default series and shapes of the NFSS, but it's not uncommon to have font families that have multiple weights and shapes and so on.

If you set up a regular font family with the 'standard four' (upright, bold, italic, and bold italic) shapes and then want to use, say, a light font for a certain document element, many users will be perfectly happy to use \newfontface\\switch\ and use the resulting font \\switch\. In other cases, however, it is more convenient or even necessary to load additional fonts using additional NFSS specifiers.

```
FontFace = \{\langle series \rangle\} \{\langle shape \rangle\} \{ Font = \langle font name \rangle, \langle features \rangle \}
FontFace = \{\langle series \rangle\} \{\langle shape \rangle\} \{\langle font name \rangle\}
```

The font thus specified will inherit the font features of the main font, with optional additional $\langle features \rangle$ as requested. (Note that the optional $\{\langle features \rangle\}$ argument is still surrounded with curly braces.) Multiple FontFace commands may be used in a single declaration to specify multiple fonts. As an example:

¹Thanks to Luca Fascione for the example and motivation for finally implementing this feature.

```
\setmainfont{font1.otf}[
  FontFace = {c}{\updefault}{ font2.otf } ,
  FontFace = {c}{m}{ Font = font3.otf , Color = red }
]
```

Writing \fontseries{c}\selectfont will result in font2 being selected, which then followed by \fontshape{m}\selectfont will result in font3 being selected (in red). A font face that is defined in terms of a different series but an upright shape (\updatupdefault, as shown above) will attempt to find a matching small caps feature and define that face as well. Conversely, a font face defined in terms of a non-standard font shape will not.

There are some standards for choosing shape and series codes; the \LaTeX $2_{\mathcal{E}}$ font selection guide² lists series m for medium, b for bold, bx for bold extended, sb for semibold, and c for condensed. A far more comprehensive listing is included in Appendix A of Philipp Lehman's 'The Font Installation Guide'³ covering 14 separate weights and 12 separate widths.

The FontFace command also interacts properly with the SizeFeatures command as follows: (nonsense set of font selection choices)

Note that if the first Font feature is omitted then each size needs its own inner Font declaration.

6.4 Math(s) fonts

When \setmainfont, \setsansfont and \setmonofont are used in the preamble, they also define the fonts to be used in maths mode inside the \mathrm-type commands. This only occurs in the preamble because LATEX freezes the maths fonts after this stage of the processing. The fontspec package must also be loaded after any maths font packages (e.g., euler) to be successful. (Actually, it is only euler that is the problem.4)

Note that fontspec will not change the font for general mathematics; only the upright and bold shapes will be affected. To change the font used for the mathematical symbols, see either the mathspec package or the unicode-math package.

Note that you may find that loading some maths packages won't be as smooth as you expect since fontspec (and X_∃T_EX in general) breaks many of the assumptions of T_EX as to where maths characters and accents can be found. Contact me if you have

²texdoc fntguide

³texdoc fontinstallationguide

 $^{^4} Speaking of euler, if you want to use its [mathbf] option, it won't work, and you'll need to put this after fontspec is loaded instead: \\ AtBeginDocument{\DeclareMathAlphabet\mathbf{U}{eur}{b}{n} }$

troubles, but I can't guarantee to be able to fix any incompatibilities. The Lucida and Euler maths fonts should be fine; for all others keep an eye out for problems.

However, the default text fonts may not necessarily be the ones you wish to use when typesetting maths (especially with the use of fancy ligatures and so on). For this reason, you may optionally use the commands above (in the same way as our other \fontspec-like commands) to explicitly state which fonts to use inside such commands as \mathrm. Additionally, the \setboldmathrm command allows you define the font used for \mathrm when in bold maths mode (which is activated with, among others, \boldmath).

For example, if you were using Optima with the Euler maths font, you might have this in your preamble:

```
\usepackage{mathpazo}
\usepackage{fontspec}
\setmainfont{Optima}
\setmathrm{Optima}
\setboldmathrm[BoldFont={Optima ExtraBlack}]{Optima Bold}
```

These commands are compatible with the unicode-math package. Having said that, unicode-math also defines a more general way of defining fonts to use in maths mode, so you can ignore this subsection if you're already using that package.

7 Miscellaneous font selecting details

The optional argument — from v2.4 For the first decade of fontspec's life, optional font features were selected with a bracketed argument before the font name, as in:

```
\setmainfont[
  lots and lots ,
  and more and more ,
  an excessive number really ,
  of font features could go here
]{myfont.otf}
```

This always looked like ugly syntax to me, because the most important detail — the name of the font — was tucked away at the end. The order of these arguments has now been reversed:

```
\setmainfont{myfont.otf}[
  lots and lots ,
  and more and more ,
  an excessive number really ,
  of font features could go here
]
```

I hope this doesn't cause any problems.

- 1. Backwards compatibility has been preserved, so either input method works.
- 2. In fact, you can write

```
\fontspec[Ligatures=Rare] {myfont.otf} [Color=red]
```

if you really felt like it and both sets of features would be applied.

3. Following standard xparse behaviour, there must be no space before the opening bracket; writing

```
\fontspec{myfont.otf}_{\sqcup}[Color=red]
```

will result in [Color=red] not being recognised an argument and therefore it will be typeset as text. When breaking over lines, write either of:

```
\fontspec{myfont.otf}% \fontspec{myfont.otf}[
    [Color=red] Color=Red]
```

Spaces \fontspec and \addfontfeatures ignore trailing spaces as if it were a 'naked' control sequence; e.g., 'M. \fontspec{...} N' and 'M. \fontspec{...}N' are the same.

Part III

Selecting font features

The commands discussed so far such as \fontspec each take an optional argument for accessing the font features of the requested font. Commands are provided to set default features to be applied for all fonts, and even to change the features that a font is presently loaded with. Different font shapes can be loaded with separate features, and different features can even be selected for different sizes that the font appears in. This part discusses these options.

8 Default settings

```
\defaultfontfeatures{\(\)font features\\}
```

It is sometimes useful to define font features that are applied to every subsequent font selection command. This may be defined with the \defaultfontfeatures command, shown in Example 5. New calls of \defaultfontfeatures overwrite previous ones, and defaults can be reset by calling the command with an empty argument.

```
\defaultfontfeatures[\langle font name \rangle] \{ \langle font features \rangle \}
```

Default font features can be specified on a per-font and per-face basis by using the optional argument to \defaultfontfeatures as shown.

```
\defaultfontfeatures[texgyreadventor-regular.otf]{Color=blue}
\setmainfont{texgyreadventor-regular.otf}% will be blue
```

Multiple fonts may be affected by using a comma separated list of font names.

```
\verb|\defaultfontfeatures[(\font-switch)]{|} {\langle font features \rangle}|
```

New in v2.4. Defaults can also be applied to symbolic families such as those created with the $\mbox{newfontfamily}$ command and for $\mbox{rmfamily}$, $\mbox{sffamily}$, and $\mbox{ttfamily}$:

```
\defaultfontfeatures[\rmfamily,\sffamily]{Ligatures=TeX}
\setmainfont{texgyreadventor-regular.otf}% will use standard TeX ligatures
```

Example 5: A demonstration of the \defaultfontfeatures command.

```
\fontspec{texgyreadventor-regular.otf}
Some default text o123456789 \\
\defaultfontfeatures{
    Numbers=OldStyle, Color=888888
}
\fontspec{texgyreadventor-regular.otf}
Now grey, with old-style figures:
    0123456789
```

Some default text 0123456789

Now grey, with old-style figures: 0123456789

The line above to set TEX-like ligatures is now activated by default in fontspec.cfg. To reset default font features, simply call the command with an empty argument:

```
\defaultfontfeatures[\rmfamily,\sffamily]{}
\setmainfont{texgyreadventor-regular.otf}% will no longer use standard TeX ligatures
```

```
\label{lem:defaultfontfeatures} $$ \defaultfontfeatures+{\langle font features \rangle} $$ \defaultfontfeatures+{\langle font name \rangle} {\langle font features \rangle} $$
```

New in v2.4. Using the + form of the command appends the $\langle font \, features \rangle$ to any already-selected defaults.

9 Default settings from a file

In addition to the defaults that may be specified in the document as described above, when a font is first loaded, a configuration file is searched for with the name '(fontname).fontspec'.5

The contents of this file can be used to specify default font features without having to have this information present within each document. \(\lambda fontname \rangle \) is stripped of spaces and file extensions are omitted; for example, the line above for TEX Gyre Adventor could be placed in a file called TeXGyreAdventor.fontspec, or for specifying options for texgyreadventor-regular.otf (when loading by filename), the configuration file would be texgyreadventor-regular.fontspec. (N.B. the lettercase of the names should match.)

This mechanism can be used to define custom names or aliases for your font collections. If you create a file MyCharis.fontspec containing, say,

```
\defaultfontfeatures[My Charis]
{
   Extension = .ttf ,
   UprightFont = CharisSILR,
   BoldFont = CharisSILB,
   ItalicFont = CharisSILI,
   BoldItalicFont = CharisSILBI,
   % <any other desired options>
}
```

you can load that custom family with \fontspec{My Charis} and similar. The optional argument to \defaultfontfeatures must match that requested by the font loading command (\fontspec, etc.), else the options won't take effect.

Finally, note that options for font faces can also be defined in this way. To continue the example above, here we colour the different faces:

```
\defaultfontfeatures[CharisSILR]{Color=blue}
\defaultfontfeatures[CharisSILB]{Color=red}
```

And such configuration lines can be stored either inline inside My Charis.fontspec or within their own .fontspec files; in this way, fontspec is designed to handle 'nested' configuration options as well.

⁵Located in the current folder or within a standard texmf location.

10 Working with the currently selected features

```
\label{lem:liftontFeatureActiveTF} $$ \left( font\ feature \right) = \left( false\ code \right) = \left( false\ code \right) $$
```

This command queries the currently selected font face and executes the appropriate branch based on whether the *\(font feature \)* as specified by fontspec is currently active. For example, the following will print 'True':

```
\setmainfont{texgyrepagella-regular.otf}[Numbers=OldStyle]
\IfFontFeatureActiveTF{Numbers=OldStyle}{True}{False}
```

Note that there is no way for fontspec to know what the default features of a font will be. For example, by default the texgyrepagella fonts use lining numbers. But in the following example, querying for lining numbers returns false since they have not been explicitly requested:

```
\setmainfont{texgyrepagella-regular.otf}
\IfFontFeatureActiveTF{Numbers=Lining}{True}{False}
```

Please note: At time of writing this function only supports OpenType fonts; AAT/Graphite fonts under the X∃TEX engine are not supported.

```
\addfontfeatures\{\langle fontfeatures \rangle\}
```

This command allows font features to be changed without knowing what features are currently selected or even what font is being used. A good example of this could be to add a hook to all tabular material to use monospaced numbers, as shown in Example 6. If you attempt to change an already-selected feature, fontspec will try to deactivate any features that clash with the new ones. *E.g.*, the following two invocations are mutually exclusive:

```
\addfontfeature{Numbers=OldStyle}...
\addfontfeature{Numbers=Lining}...
```

Since Numbers=Lining comes last, it takes precedence and deactivates the call Numbers=OldStyle. This command may also be executed under the alias \addfontfeature.

\addfontfeature

10.1 Priority of feature selection

Features defined with \addfontfeatures override features specified by \fontspec , which in turn override features specified by \adfaultfontfeatures . If in doubt, whenever a new font is chosen for the first time, an entry is made in the transcript (.log) file displaying the font name and the features requested.

Example 6: A demonstration of the \addfontfeatures command.

'In 1842, 999 people sailed 97 miles in 13 boats. In 1923, 111 people sailed 54 miles in 56 boats.'

```
        Year
        People
        Miles
        Boats

        1842
        999
        75
        13

        1923
        111
        54
        56
```

11 Different features for different font shapes

```
BoldFeatures={\langle features \rangle}
ItalicFeatures={\langle features \rangle}
BoldItalicFeatures={\langle features \rangle}
SlantedFeatures={\langle features \rangle}
BoldSlantedFeatures={\langle features \rangle}
SmallCapsFeatures={\langle features \rangle}
```

It is entirely possible that separate fonts in a family will require separate options; *e.g.*, Hoefler Text Italic contains various swash feature options that are completely unavailable in the upright shapes.

The font features defined at the top level of the optional \fontspec argument are applied to all shapes of the family. Using Upright-, SmallCaps-, Bold-, Italic-, and BoldItalicFeatures, separate font features may be defined to their respective shapes in addition to, and with precedence over, the 'global' font features. See Example 7.

Note that because most fonts include their small caps glyphs within the main font, features specified with SmallCapsFeatures are applied in addition to any other shape-specific features as defined above, and hence SmallCapsFeatures can be nested within ItalicFeatures and friends. Every combination of upright, italic, bold and small caps can thus be assigned individual features, as shown in the somewhat ludicrous Example 8.

Example 7: Features for, say, just italics.		
Don't Ask Victoria! Don't Ask Victoria!	\fontspec{EBGaramond12-Regular.otf}% [ItalicFont=EBGaramond12-Italic.otf] \itshape Don't Ask Victoria! \\ \addfontfeature{ItalicFeatures={Style=Swash}} Don't Ask Victoria! \\	

```
\fontspec{texgyretermes}[
                                      Extension = {.otf},
                                      UprightFont = {*-regular}, ItalicFont = {*-italic},
                                      BoldFont = {*-bold}, BoldItalicFont = {*-bolditalic},
                                      UprightFeatures={Color = 220022,
                                           SmallCapsFeatures = {Color=115511}},
                                       ItalicFeatures={Color = 2244FF,
                                           SmallCapsFeatures = {Color=112299}},
                                         BoldFeatures={Color = FF4422,
                                           SmallCapsFeatures = {Color=992211}},
                                   BoldItalicFeatures={Color = 888844,
                                           SmallCapsFeatures = {Color=444422}},
Upright Small Caps
                                  Upright {\scshape Small Caps}\\
Italic Italic Small Caps
                                  \itshape Italic {\scshape Italic Small Caps}\\
Bold Bold Small Caps
                                  \upshape\bfseries Bold {\scshape Bold Small Caps}\\
Bold Italic Bold Italic Small Caps
                                  \itshape Bold Italic {\scshape Bold Italic Small Caps}
```

Selecting fonts from TrueType Collections (TTC files)

TrueType Collections are multiple fonts contained within a single file. Each font within a collection must be explicitly chosen using the FontIndex command. Since TrueType Collections are often used to contain the italic/bold shapes in a family, fontspec automatically selects the italic, bold, and bold italic fontfaces from the same file. For example, to load the macOS system font Optima:

```
\setmainfont{Optima.ttc}[
  Path = /System/Library/Fonts/ ,
  UprightFeatures = {FontIndex=0} ,
  BoldFeatures = {FontIndex=1} ,
  ItalicFeatures = {FontIndex=2} ,
  BoldItalicFeatures = {FontIndex=3} ,
]
```

Support for TrueType Collections has only been tested in X₄T_EX, but should also work with an up-to-date version of LuaT_EX and the luaotfload package.

13 Different features for different font sizes

```
SizeFeatures = {
    ...
    { Size = \langle size range \rangle, \langle font features \rangle },
    { Size = \langle size range \rangle, \text{Font} = \langle font name \rangle, \langle font features \rangle },
    ...
}
```

The SizeFeature feature is a little more complicated than the previous features discussed. It allows different fonts and different font features to be selected for a given font family as the point size varies.

It takes a comma separated list of braced, comma separated lists of features for each size range. Each sub-list must contain the Size option to declare the size range, and optionally Font to change the font based on size. Other (regular) fontspec features that are added are used on top of the font features that would be used anyway. A demonstration to clarify these details is shown in Example 9. A less trivial example is shown in the context of optical font sizes in Section 14.6 on page 30.

To be precise, the Size sub-feature accepts arguments in the form shown in Table I on the next page. Braces around the size range are optional. For an exact font size (Size=X) font sizes chosen near that size will 'snap'. For example, for size definitions at exactly IIPt and I4Pt, if a I2Pt font is requested actually the IIPt font will be selected. This is a remnant of the past when fonts were designed in metal (at obviously rigid sizes) and later when bitmap fonts were similarly designed for fixed sizes.

If additional features are only required for a single size, the other sizes must still be specified. As in:

```
SizeFeatures={
    {Size=-10,Numbers=Uppercase},
    {Size=10-}}
```

Otherwise, the font sizes greater than 10 won't be defined at all!

Interaction with other features For SizeFeatures to work with ItalicFeatures,
BoldFeatures, etc., and SmallCapsFeatures, a strict heirarchy is required:

```
UprightFeatures =
{
```

Example 9: An example of specifying different font features for different sizes of font with Size-Features.

Table 1: Syntax for specifying the size to apply custom font features.

Input	Font size, s
Size = X-	$s \geq X$
Size = -Y	$s < \mathbf{Y}$
Size = X-	$\mathbf{Y} \mathbf{X} \leq s < \mathbf{Y}$
Size = X	$s={\tt X}$

```
SizeFeatures =
{
      {
          Size = -10,
          Font = ..., % if necessary
          SmallCapsFeatures = {...},
          ... % other features for this size range
      },
          ... % other size ranges
    }
}
```

Suggestions on simplifying this interface welcome.

14 Font independent options

Features introduced in this section may be used with any font.

14.1 Colour

Color (or Colour) uses font specifications to set the colour of the text. You should think of this as the literal glyphs of the font being coloured in a certain way. Notably, this mechanism is different to that of the color/xcolor/hyperref/etc. packages, and in fact using fontspec commands to set colour will prevent your text from changing colour using those packages at all! For example, if you set the colour in a \setmainfont command, \color{...} and related commands, including hyperlink colouring, will no longer have any effect on text in this font.) Therefore, fontspec's colour commands are best used to set explicit colours in specific situations, and the xcolor package is recommended for more general colour functionality.

The colour is defined as a triplet of two-digit Hex RGB values, with optionally another value for the transparency (where oo is completely transparent and FF is opaque.) Transparency is supported by Lual-TeX; Xal-TeX with the xdvipdfmx driver does not support this feature.

If you load the xcolor package, you may use any named colour instead of writing the colours in hexadecimal.

```
\usepackage{xcolor}
...
\fontspec[Color=red]{Verdana} ...
```

Example 10: Selecting colour with transparency.



\fontsize{48}{48} \fontspec{texgyrebonum-bold.otf} {\addfontfeature{Color=FF000099}W}\kern-o.4ex {\addfontfeature{Color=ooooFF99}S}\kern-o.4ex {\addfontfeature{Color=DDBB2299}P}\kern-o.5ex {\addfontfeature{Color=ooBB3399}R}

```
\definecolor{Foo}{rgb}{0.3,0.4,0.5}
\fontspec[Color=Foo]{Verdana} ...
```

The color package is not supported; use xcolor instead.

You may specify the transparency with a named colour using the Opacity feature which takes an decimal from zero to one corresponding to transparent to opaque respectively:

```
\fontspec[Color=red,Opacity=0.7]{Verdana} ...
```

It is still possible to specify a colour in six-char hexadecimal form while defining opacity in this way, if you like.

Scale 14.2

```
Scale = \langle number \rangle
Scale = MatchLowercase
Scale = MatchUppercase
```

In its explicit form, Scale takes a single numeric argument for linearly scaling the font, as demonstrated in Example 1. It is now possible to measure the correct dimensions of the fonts loaded and calculate values to scale them automatically.

As well as a numerical argument, the Scale feature also accepts options MatchLowercase and MatchUppercase, which will scale the font being selected to match the current default roman font to either the height of the lowercase or uppercase letters, respectively; these features are shown in Example II.

The amount of scaling used in each instance is reported in the .log file. Since there is some subjectivity about the exact scaling to be used, these values should be used to fine-tune the results.

Example 11: Automatically calculated scale values.

\setmainfont{Georgia} \newfontfamily\lc[Scale=MatchLowercase]{Verdana} The perfect match {\lc is hard to find.}\\ $The \ perfect \ match \ is \ hard \ to \ find. \quad \verb|\newfontfamily| uc[Scale=MatchUppercase]{Arial}|$ LOGO\uc FON T

LOGOFONT

Note that when Scale=MatchLowercase is used with \setmainfont, the new 'main' font of the document will be scaled to match the old default. This may be undesirable in some cases, so to achieve 'natural' scaling for the main font but automatically scale all other fonts selected, you may write

```
\defaultfontfeatures{ Scale = MatchLowercase }
\defaultfontfeatures[\rmfamily]{ Scale = 1}
```

One or both of these lines may be placed into a local fontspec.cfg file (see Section 3.3 on page 7) for this behaviour to be effected in your own documents automatically. (Also see Section 8 on page 20 for more information on setting font defaults.)

14.3 Interword space

While the space between words can be varied on an individual basis with the TEX primitive \spaceskip command, it is more convenient to specify this information when the font is first defined.

The space in between words in a paragraph will be chosen automatically, and generally will not need to be adjusted. For those times when the precise details are important, the WordSpace feature is provided, which takes either a single scaling factor to scale the default value, or a triplet of comma-separated values to scale the nominal value, the stretch, and the shrink of the interword space by, respectively. (WordSpace= $\{x\}$ is the same as WordSpace= $\{x,x,x\}$.)

Note that TEX's optimisations in how it loads fonts means that you cannot use this feature in \addfontfeatures.

14.4 Post-punctuation space

If \frenchspacing is not in effect, TeX will allow extra space after some punctuation in its goal of justifying the lines of text. Generally, this is considered old-fashioned, but occasionally in small amounts the effect can be justified, pardon the pun.

The PunctuationSpace feature takes a scaling factor by which to adjust the nominal value chosen for the font; this is demonstrated in Example 13. Note that PunctuationSpace=0 is not equivalent to \frenchspacing, although the difference will only be apparent when a line of text is under-full.

Note that TEX's optimisations in how it loads fonts means that you cannot use this feature in \addfontfeatures.

14.5 The hyphenation character

The letter used for hyphenation may be chosen with the HyphenChar feature. It takes three types of input, which are chosen according to some simple rules. If the input is the string None, then hyphenation is suppressed for this font. If the input is a single character, then this character is used. Finally, if the input is longer than a single character it must be the UTF-8 slot number of the hyphen character you desire.

This package redefines LaTeX's \- macro such that it adjusts along with the above changes.

Note that TEX's optimisations in how it loads fonts means that you cannot use this feature in \addfontfeatures.

Example 12: Scaling the default interword space. An exaggerated value has been chosen to emphasise the effects here.

> \fontspec{texgyretermes-regular.otf} Some text for our example to take up some space, and to demonstrate the default interword space. \bigskip

Some text for our example to take up some space, and to demonstrate the default interword space.

Sometextforour example to take up some space, and to demonstrate the default interword space.

\fontspec{texgyretermes-regular.otf}% [WordSpace = 0.3] Some text for our example to take up some space, and to demonstrate the default interword space.

Example 13: Scaling the default post-punctuation space.

\nonfrenchspacing

\fontspec{texgyreschola-regular.otf}

Letters, Words. Sentences. \par

\fontspec{texgyreschola-regular.otf}[PunctuationSpace=2]

Letters, Words. Sentences. Letters, Words. Sentences. \par

Letters, Words. Sentences. \fontspec{texgyreschola-regular.otf}[PunctuationSpace=o]

Letters, Words. Sentences. Letters, Words. Sentences.

Example 14: Explicitly choosing the hyphenation character.

EXAMPLE HYPHENATION	<pre>\def\text{\fbox{\parbox{1.55cm}{% EXAMPLE HYPHENATION% }}\qquad\qquad\null\par\bigskip}</pre>
EXAMPLE HYPHEN±	\fontspec{Linux Libertine 0}[HyphenChar=None] \text

\fontspec{Linux Libertine 0}[HyphenChar={+}] ATION

\text

14.6 Optical font sizes

Optically scaled fonts thicken out as the font size decreases in order to make the glyph shapes more robust (less prone to losing detail), which improves legibility. Conversely, at large optical sizes the serifs and other small details may be more delicately rendered.

OpenType fonts with optical scaling will exist in several discrete sizes, and these will be selected by XaTeX and LuaTeX automatically determined by the current font size as in Example 15, in which we've scaled down some large text in order to be able to compare the difference for equivalent font sizes.

The OpticalSize option may be used to specify a different optical size. With OpticalSize set to zero, no optical size font substitution is performed, as shown in Example 16.

The SizeFeatures feature (Section 13 on page 25) can be used to specify exactly which optical sizes will be used for ranges of font size. For example, something like:

14.7 Font transformations

In rare situations users may want to mechanically distort the shapes of the glyphs in the current font such as shown in Example 17. Please don't overuse these features; they are not a good alternative to having the real shapes.

If values are omitted, their defaults are as shown above.

If you want the bold shape to be faked automatically, or the italic shape to be slanted automatically, use the AutoFakeBold and AutoFakeSlant features. For example, the following two invocations are equivalent:

```
\fontspec[AutoFakeBold=1.5]{Charis SIL}
\fontspec[BoldFeatures={FakeBold=1.5}]{Charis SIL}
```

If both of the AutoFake... features are used, then the bold italic font will also be faked.

The FakeBold and AutoFakeBold features are only available with the X₃T_EX engine and will be ignored in LuaT_EX.

Example 15: A demonstration of automatic optical size selection.		
	\fontspec{Latin Modern Roman} Automatic optical size	\\
Automatic optical size Automatic optical size	\scalebox{o.4}{\Huge Automatic optical size}	

Example 16: Optical size substitution is suppressed when set to zero.

	Latin Modern Roman Latin Modern optical sizes	5 Regular}[OpticalSize=o] \\
	Latin Modern Roman	8 Regular}[OpticalSize=o]
T N.T 1 1	Latin Modern optical sizes	\\
Latin Modern optical sizes	Latin Modern Roman	<pre>12 Regular}[OpticalSize=o]</pre>
Latin Modern optical sizes	Latin Modern optical sizes	\\
Latin Modern optical sizes	Latin Modern Roman	<pre>17 Regular}[OpticalSize=o]</pre>
Latin Modern optical sizes	Latin Modern optical sizes	

	Example	17: Articifial font transformations.
		\fontspec{Charis SIL} \emph{ABCxyz} \fontspec{Charis SIL}[FakeSlant=0.2] ABCxyz
ABCxyz	ABCxyz	\fontspec{Charis SIL} ABCxyz \fontspec{Charis SIL}[FakeStretch=1.2] ABCxyz
•	ABCxyz ABCxyz	\fontspec{Charis SIL} \textbf{ABCxyz} \fontspec{Charis SIL}[FakeBold=1.5] ABCxyz

14.8 Letter spacing

Letter spacing, or tracking, is the term given to adding (or subtracting) a small amount of horizontal space in between adjacent characters. It is specified with the LetterSpace, which takes a numeric argument, shown in Example 18.

The letter spacing parameter is a normalised additive factor (not a scaling factor); it is defined as a percentage of the font size. That is, for a 10 pt font, a letter spacing parameter of '1.0' will add 0.1 pt between each letter.

This functionality is not generally used for lowercase text in modern typesetting but does have historic precedent in a variety of situations. In particular, small amounts of letter spacing can be very useful, when setting small caps or all caps titles. Also see the OpenType Uppercase option of the Letters feature (Section 16.2 on page 34).

| Example 18: The LetterSpace feature. | \fontspec{Didot} \addfontfeature{LetterSpace=0.0} \use tracking for display caps text \use tracking for display caps text

Part IV

OpenType

15 Introduction

OpenType fonts (and other 'smart' font technologies such as AAT and Graphite) can change the appearance of text in many different ways. These changes are referred to as font features. When the user applies a feature — for example, small capitals — to a run of text, the code inside the font makes appropriate substitutions and small capitals appear in place of lowercase letters. However, the use of such features does not affect the underlying text. In our small caps example, the lowercase letters are still stored in the document; only the appearance has been changed by the OpenType feature. This makes it possible to search and copy text without difficulty. If the user selected a different font that does not support small caps, the 'plain' lowercase letters would appear instead.

Some OpenType features are required to support particular scripts, and these features are often applied automatically. The Indic scripts, for example, often require that characters be reshaped and reordered after they are typed by the user, in order to display them in the traditional ways that readers expect. Other features can be applied to support a particular language. The Junicode font for medievalists uses by default the Old English shape of the letter thorn, while in modern Icelandic thorn has a more rounded shape. If a user tags some text as being in Icelandic, Junicode will automatically change to the Icelandic shape through an OpenType feature that localises the shapes of letters.

There are a large group of OpenType features, designed to support high quality typography a multitude of languages and writing scripts. Examples of some font features have already been shown in previous sections; the complete set of OpenType font features supported by fontspec is described below in Section 16.

The OpenType specification provides four-letter codes (e.g., smcp for small capitals) for each feature. The four-letter codes are given below along with the fontspec names for various features, for the benefit of people who are already familiar with OpenType. You can ignore the codes if they don't mean anything to you.

15.1 How to select font features

Font features are selected by a series of $\langle feature \rangle = \langle option \rangle$ selections. Features are (usually) grouped logically; for example, all font features relating to ligatures are accessed by writing Ligatures={...} with the appropriate argument(s), which could be TeX, Rare, etc., as shown below in 16.1.1.

Multiple options may be given to any feature that accepts non-numerical input, although doing so will not always work. Some options will override others in generally obvious ways; Numbers={OldStyle,Lining} doesn't make much sense because the two options are mutually exclusive, and XaTeX will simply use the last option that is specified (in this case using Lining over OldStyle).

If a feature or an option is requested that the font does not have, a warning is given in the console output. As mentioned in Section 3.4 on page 8 these warnings can be

suppressed by selecting the [quiet] package option.

15.2 How do I know what font features are supported by my fonts?

Although I've long desired to have a feature within fontspec to display the OpenType features within a font, it's never been high on my priority list. One reason for that is the existence of the document opentype-info.tex, which is available on CTAN or typing kpsewhich opentype-info.tex in a Terminal window. Make a copy of this file and place it somewhere convenient. Then open it in your regular TeX editor and change the font name to the font you'd like to query; after running through plain XaTeX, the output PDF will look something like this:

```
OpenType Layout features found in '[Asana-Math.otf]'
script = 'DFLT'
    \mathsf{language} = \langle \mathsf{default} \rangle
        features = 'onum' 'salt' 'kern'
script = 'cher'
    language = \langle default \rangle
        features = 'onum' 'salt' 'kern'
script = 'grek'
    language = \langle default \rangle
        features = 'onum' 'salt' 'kern'
script = 'latn'
    language = \langle default \rangle
        features = 'onum' 'salt' 'kern'
script = 'math'
    language = \langle default \rangle
        features = 'dtls' 'onum' 'salt' 'ssty' 'kern'
```

I intentionally picked a font that by design needs few font features; 'regular' text fonts such as Latin Modern Roman contain many more, and I didn't want to clutter up the document too much. You'll then need to cross-check the OpenType feature tags with the 'logical' names used by fontspec.

otfinfo Alternatively, and more simply, you can use the command line tool otfinfo, which is distributed with TEXLive. Simply type in a Terminal window, say:

```
otfinfo -f `kpsewhich lmromandunh1o-oblique.otf`
```

which results in:

```
aalt Access All Alternates
cpsp Capital Spacing
dlig Discretionary Ligatures
frac Fractions
```

kern	Kerning			
liga	Standard Ligatures			
lnum	Lining Figures			
onum	Oldstyle Figures			
pnum	Proportional Figures			
size	Optical Size			
tnum	Tabular Figures			
zero	Slashed Zero			

16 OpenType font features

There are a finite set of OpenType font features, and fontspec provides an interface to around half of them. Full documentation will be presented in the following sections, including how to enable and disable individual features, and how they interact.

A brief reference is provided (Table 2 on the next page) but note that this is an incomplete listing — only the 'enable' keys are shown, and where alternative interfaces are provided for convenience only the first is shown. (E.g., Numbers=OldStyle is the same as Numbers=Lowercase.)

For completeness, the complete list of OpenType features not provided with a fontspec interface is shown in Table 3 on page 36. Features omitted are partially by design and partially by oversight; for example, the aalt feature is largely useless in TEX since it is designed for providing a textscgui interface for selecting 'all alternates' of a glyph. Others, such as optical bounds for example, simply haven't yet been considered due to a lack of fonts available for testing. Suggestions welcome for how/where to add these missing features to the package.

16.1 Tag-based features

16.1.1 Ligatures

Ligatures refer to the replacement of two separate characters with a specially drawn glyph for functional or æsthetic reasons. The list of options, of which multiple may be selected at one time, is shown in Table 4. A demonstration with the Linux Libertine fonts⁶ is shown in Example 19.

Note the additional features accessed with Ligatures=TeX. These are not actually real OpenType features, but additions provided by luaotfload (i.e., LuaTeX only) to emulate TeX's behaviour for ASCII input of curly quotes and punctuation. In XaTeX this is achieved with the Mapping feature (see Section 21.1 on page 59) but for consistency Ligatures=TeX will perform the same function as Mapping=tex-text.

16.2 Letters

The Letters feature specifies how the letters in the current font will look. OpenType fonts may contain the following options: Uppercase, SmallCaps, PetiteCaps, UppercaseSmallCaps, UppercasePetiteCaps, and Unicase.

⁶http://www.linuxlibertine.org/

Table 2: Summary of OpenType features in fontspec, alphabetic by feature tag.

ABVM	Diacritics = AboveBase	Above-base Mark	NUMR	VerticalPosition = Numerator	Numerators
		Positioning	ONUM	Numbers = Lowercase	Oldstyle Figures
AFRC	Fractions = Alternate	Alternative Fractions	ORDN	VerticalPosition = Ordinal	Ordinals
BLWM	Diacritics = BelowBase	Below-base Mark	ORNM	Ornament = N	Ornaments
CALT	Contextuals = Alternate	Positioning Contextual Alternates	PALT	CharacterWidth = AlternateProportional	Proportional Alternate Widths
CASE	Letters = Uppercase	Case-Sensitive Forms	PCAP	Letters = PetiteCaps	Petite Capitals
CLIG	Ligatures = Contextual	Contextual Ligatures	PKNA	Style = ProportionalKana	Proportional Kana
CPSP	Kerning = Uppercase	Capital Spacing	PNUM	Numbers = Proportional	Proportional Figures
CSWH	Contextuals = Swash	Contextual Swash	PWID	CharacterWidth = Proportional	Proportional Widths
cvNN	${\tt CharacterVariant=} N\!:\! M$	Character Variant N	QWID	CharacterWidth = Quarter	Quarter Widths
C2PC	Letters = UppercasePetiteCaps	Petite Capitals From	RAND	Letters = Random	Randomize
		Capitals	RLIG	Ligatures = Required	Required Ligatures
C2SC	Letters = UppercaseSmallCaps	Small Capitals From	RUBY	Style = Ruby	Ruby Notation Forms
		Capitals	SALT	Alternate = N	Stylistic Alternates
DLIG	Ligatures = Rare	Discretionary Ligatures	SINF	VerticalPosition = ScientificInferior	Scientific Inferiors
DNOM		Denominators	SMCP	Letters = SmallCaps	Small Capitals
EXPT	CJKShape = Expert	Expert Forms	SMPL	CJKShape = Simplified	Simplified Forms
FALT	Contextuals = LineFinal	Final Glyph on Line Alternates	ssNN	StylisticSet = N	Stylistic Set N
FINIA	Contextuals = WordFinal	Terminal Forms	SSTY	Style = MathScript	Math script style alternates
FINA FRAC	Fractions = On	Fractions	SUBS	VerticalPosition = Inferior	Subscript
FWID	CharacterWidth = Full	Full Widths	SUPS	VerticalPosition = Superior	Superscript
HALT	CharacterWidth = AlternateHalf	Alternate Half Widths	SWSH	Style = Swash	Swash
HIST	Style = Historic	Historical Forms	TITL	Style = TitlingCaps	Titling
HKNA	Style = HorizontalKana	Horizontal Kana Alternates	TNUM	Numbers = Monospaced	Tabular Figures
HLIG	Ligatures = Historic	Historical Ligatures	TRAD	CJKShape = Traditional	Traditional Forms
HWID	CharacterWidth = Half	Half Widths	TWID	CharacterWidth = Third	Third Widths
INIT	Contextuals = WordInitial	Initial Forms	UNIC	Letters = Unicase	Unicase
ITAL	Style = Italic	Italics	VALT	Vertical = AlternateMetrics	Alternate Vertical Metrics
JP78	CJKShape = JIS1978	JIS78 Forms	VERT	Vertical = Alternates	Vertical Writing
JP83	CJKShape = JIS1983	JIS83 Forms	VHAL	Vertical = HalfMetrics	Alternate Vertical Half
JP90	CJKShape = JIS1990	JIS90 Forms			Metrics
JP04	CJKShape = JIS2004	JIS2004 Forms	VKNA	Style = VerticalKana	Vertical Kana Alternates
KERN	Kerning = On	Kerning	VKRN	Vertical = Kerning	Vertical Kerning
LIGA	Ligatures = Common	Standard Ligatures	VPAL	Vertical = ProportionalMetrics	Proportional Alternate Vertical Metrics
LNUM	Numbers = Uppercase	Lining Figures	VRT2	Vertical = RotatedGlyphs	Vertical Alternates and
MARK	Diacritics = MarkToBase	Mark Positioning	V IX 12	. or	Rotation
MEDI	Contextuals = Inner	Medial Forms	VRTR	Vertical = AlternatesForRotation	Vertical Alternates for
MKMK	Diacritics = MarkToMark	Mark to Mark Positioning			Rotation
NALT	Annotation = N	Alternate Annotation Forms	ZERO	Numbers = SlashedZero	Slashed Zero
NLCK	CJKShape = NLC	NLC Kanji Forms			

 $\label{thm:continuous} \mbox{Table 3: List of unsupported OpenType features.}$

AALT Access All Alternates	HNGL Hangul	PSTS Post-base Substitutions
ABVF Above-base Forms	нојо <i>Hojo Kanji Forms</i>	RCLT Required Contextual
ABVS Above-base Substitutions	ISOL Isolated Forms	Alternates
akhn Akhands	JALT Justification Alternates	RKRF Rakar Forms
BLWF Below-base Forms	LFBD Left Bounds	RPHF Reph Forms
BLWS Below-base Substitutions	цмо Leading Jamo Forms	RTBD Right Bounds
ссмр Glyph Composition /	LOCL Localized Forms	RTLA Right-to-left alternates
Decomposition	LTRA Left-to-right alternates	RTLM Right-to-left mirrored
CFAR Conjunct Form After Ro	LTRM Left-to-right mirrored	forms
CJCT Conjunct Forms	forms	RVRN Required Variation
срст Centered CJK Punctuation	MED2 Medial Forms #2	Alternates
curs Cursive Positioning	мgrк Mathematical Greek	SIZE Optical size
DIST Distances	мseт Mark Positioning via	sтсн Stretching Glyph
OTLS Dotless Forms	Substitution	Decomposition
FIN2 Terminal Forms #2	NUKT Nukta Forms	тумо Trailing Jamo Forms
FIN3 Terminal Forms #3	орво Optical Bounds	TNAM Traditional Name Forms
FLAC Flattened accent forms	PREF Pre-Base Forms	VATU Vattu Variants
HALF Half Forms	PRES Pre-base Substitutions	vjмо Vowel Jamo Forms
HALN Halant Forms	PSTF Post-base Forms	

Table 4: Options for the OpenType font feature 'Ligatures'.

Feature	Option	Tag
Ligatures =	Required	rlig †
	Common	liga †
	Contextual	clig †
	Rare/Discretionary	dlig †
	Historic	hlig †
	TeX	tlig †
	ResetAll	

 $[\]dagger$ These feature options can be disabled with . . Off variants, and reset to default state (neither explicitly on nor off) with . . Reset.

Example 19: An example of the Ligatures feature.

 $strict \rightarrow strict$ $wurtzite \rightarrow wurtzite$ $firefly \rightarrow firefly$

\def\test#1#2{%
 #2 \$\to\$ {\addfontfeature{#1} #2}\\}
\fontspec{Linux Libertine 0}
\test{Ligatures=Historic}{strict}
\test{Ligatures=Rare}{wurtzite}
\test{Ligatures=NoCommon}{firefly}

Table 5: Options for the OpenType font feature 'Letters'.

Feature	Option	Tag	
Letters =	Uppercase	case	+
	SmallCaps	smcp	†
	PetiteCaps	pcap	†
	UppercaseSmallCaps	C2SC	†
	Uppercase Petite Caps	с2рс	†
	Unicase	unic	†
	ResetAll		

[†] These feature options can be disabled with . .Off variants, and reset to default state (neither explicitly on nor off) with . .Reset.

Petite caps are smaller than small caps. SmallCaps and PetiteCaps turn lowercase letters into the smaller caps letters, whereas the Uppercase... options turn the capital letters into the smaller caps (good, *e.g.*, for applying to already uppercase acronyms like 'NASA'). This difference is shown in Example 20. 'Unicase' is a weird hybrid of upper and lower case letters.

Note that the Uppercase option will (probably) not actually map letters to uppercase.⁷ It is designed to select various uppercase forms for glyphs such as accents and dashes, such as shown in Example 21; note the raised position of the hyphen to better match the surrounding letters.

The Kerning feature also contains an Uppercase option, which adds a small amount of spacing in between letters (see Section 16.5 on page 43).

16.2.1 Numbers

The Numbers feature defines how numbers will look in the selected font, accepting options shown in Table 6.

The synonyms Uppercase and Lowercase are equivalent to Lining and Old-Style, respectively. The differences have been shown previously in Section 10 on page 22. The Monospaced option is useful for tabular material when digits need to be vertically aligned.

The SlashedZero option replaces the default zero with a slashed version to prevent confusion with an uppercase 'O', shown in Example 22.

The Arabic option (with tag anum) maps regular numerals to their Arabic script or Persian equivalents based on the current Language setting (see Section 16.9 on page 48). This option is based on a LuaTeX feature of the luaotfload package, not an OpenType feature. (Thus, this feature is unavailable in XaTeX.)

16.2.2 Contextuals

This feature refers to substitutions of glyphs that vary 'contextually' by their relative position in a word or string of characters; features such as contextual swashes are accessed via the options shown in Table 7.

Historic forms are accessed in OpenType fonts via the feature Style=Historic; this is generally not contextual in OpenType, which is why it is not included in this feature.

```
| Example 20: Small caps from lowercase or uppercase letters.

| 'fontspec{texgyreadventor-regular.otf} [Letters=SmallCaps] |
| THIS SENTENCE no verb | \
| THIS SENTENCE NO VERB | \fontspec{texgyreadventor-regular.otf} [Letters=UppercaseSmallCaps] |
| THIS SENTENCE no verb | THIS SENTENCE no verb |
```

Example 21: An example of the U	ppercase option of the Letters feature.
UPPER-CASE example UPPER-CASE example	\fontspec{Linux Libertine 0} UPPER-CASE example \\ \addfontfeature{Letters=Uppercase} UPPER-CASE example

Table 6: Options for the OpenType font feature 'Numbers'.

Feature	Option	Tag	
Numbers =	Uppercase	lnum	†
	Lowercase	onum	†
	Lining	lnum	†
	OldStyle	onum	†
	Proportional	pnum	†
	Monospaced	tnum	†
	SlashedZero	zero	†
	Arabic	anum	†
	ResetAll		

 $[\]dagger$ These feature options can be disabled with . . Off variants, and reset to default state (neither explicitly on nor off) with . . Reset.

	Example 22:	The effect	of the Sla	shedZero	option
--	-------------	------------	------------	----------	--------

\fontspec[Numbers=Lining]{texgyrebonum-regular.otf}
0123456789

0123456789 0123456789

\fontspec[Numbers=SlashedZero]{texgyrebonum-regular.otf}
0123456789

Table 7: Options for the OpenType font feature 'Contextuals'.

Feature	Option	Tag	
Contextuals =	Swash	cswh	†
	Alternate	calt	†
	WordInitial	init	†
	WordFinal	fina	†
	LineFinal	falt	†
	Inner	medi	†
	ResetAll		

 $[\]dagger$ These feature options can be disabled with . . Off variants, and reset to default state (neither explicitly on nor off) with . . Reset.

Table 8: Options for the OpenType font feature 'VerticalPosition'.

Feature	Option	Tag	
VerticalPosition =	Superior	sups	†
	Inferior	subs	+
	Numerator	numr	+
	Denominator	${\tt dnom}$	+
	ScientificInferior	sinf	+
	Ordinal	ordn	†
	ResetAll		

[†] These feature options can be disabled with . .Off variants, and reset to default state (neither explicitly on nor off) with . .Reset.

16.2.3 Vertical Position

The VerticalPosition feature is used to access things like subscript (Inferior) and superscript (Superior) numbers and letters (and a small amount of punctuation, sometimes). The Ordinal option will only raise characters that are used in some languages directly after a number. The ScientificInferior feature will move glyphs further below the baseline than the Inferior feature. These are shown in Example 23

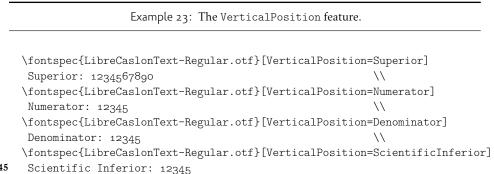
Numerator and Denominator should only be used for creating arbitrary fractions (see next section).

The realscripts package (which is also loaded by xltxtra for $X_{\overline{A}}T_{\overline{E}}X$) redefines the \textsubscript and \textsuperscript commands to use the above font features automatically, including for use in footnote labels. If this is the only feature of xltxtra you wish to use, consider loading realscripts on its own instead.

16.2.4 Fractions

For OpenType fonts use a regular text slash to create fractions, but the Fraction feature must be explicitly activated. Some (Asian fonts predominantly) also provide for the Alternate feature. These are both shown in Example 24.

 $^{^7\}mbox{If you want automatic uppercase letters, look to $\mbox{\sc LME}$X's \mbox{\sc MakeUppercase command.}}$



Superior: ¹²³⁴⁵⁶⁷⁸⁹⁰ Numerator: ¹²³⁴⁵ Denominator: ₁₂₃₄₅ Scientific Inferior: ₁₂₃₄₅

Table 9: Options for the OpenType font feature 'Fractions'.

Feature	Option	Tag	
Fractions =	On Off Reset	+frac -frac	
	Alternate	afrc ·	†
	ResetAll		

[†] These feature options can be disabled with . .Off variants, and reset to default state (neither explicitly on nor off) with . .Reset.

Example 24:	The Fractions	feature.
-------------	---------------	----------

	\fontspec{Hiragino Maru Gothic Pro W4}
1/0 1/4 5/0 10570/0400	1/2 1/4 5/6 13579/24680 \\
1/2 1/4 5/6 13579/2468	O \addfontfeature{Fractions=On}
½ ¼ % 13579/24680	1/2 1/4 5/6 13579/24680
$\frac{1}{2}$ $\frac{1}{4}$ $\frac{5}{6}$ 13579/24680	\addfontfeature{Fractions=Alternate}
	1/2 1/4 5/6 13579/24680 \\

16.3 Style

Ruby' refers to a small optical size, used in Japanese typography for annotations. For fonts with multiple salt OpenType features, use the fontspec Alternate feature instead.

Example 25 and Example 26 both contain glyph substitutions with similar characteristics. Note the occasional inconsistency with which font features are labelled; a long-tailed 'Q' could turn up anywhere!

In other features, larger breadths of changes can be seen, covering the style of an entire alphabet. See Example 27 and Example 28; in the latter, the Italic option affects the Latin text and the Ruby option the Japanese.

Note the difference here between the default and the horizontal style kana in Example 29: the horizontal style is slightly wider.

Example 25:	Example of the A	lternate option of the Style fo	eature.
	Q W Q W	Quattro M Q W S M Q W	\\

Table 10: Options for the OpenType font feature 'Style'.

Feature	Option	Tag	
Style =	Alternate	salt	†
•	Italic	ital	†
	Ruby	ruby	†
	Swash	swsh	†
	Cursive	curs	†
	Historic	hist	†
	TitlingCaps	titl	†
	Horizontal Kana	hkna	†
	VerticalKana	vkna	†
	ResetAll		

 $[\]dagger$ These feature options can be disabled with . . Off variants, and reset to default state (neither explicitly on nor off) with . . Reset.

$M \cap Z$	\fontspec{Adobe Jenson Pro}	
MQZ	M Q Z	' / /
MQZ	Style=Histor M Q Z	ic}
TITI INC. CARC	\fontspec{Adobe Garamond Pro} TITLING CAPS	\ \
TITLING CAPS	\addfontfeature{Style=TitlingCaps}	//
TITLING CAPS	TITLING CAPS	

Latin ようこそ ワカヨタレソ *Latin* ようこそ ワカヨタレソ

\fontspec{Hiragino Mincho Pro}
Latin \kana \\
\addfontfeature{Style={Italic, Ruby}}
Latin \kana

 $\label{thm:contact} \textbf{Example 29: Example of the Horizontal} \textbf{Kana and Vertical} \textbf{Kana options of the Style feature.}$

	\fontspec{Hiragino Mincho Pro}
ようこそ ワカヨタレソ	\kana \\
よりこそりカコグレク	{\addfontfeature{Style=HorizontalKana}
ようこそ ワカヨタレソ	\kana } \\
	{\addfontfeature{Style=VerticalKana}
ようこそ ワカヨタレソ	\kana }

16.4 Diacritics

Specifies how combining diacritics should be placed. These will usually be controlled automatically according to the Script setting.

16.5 Kerning

Specifies how inter-glyph spacing should behave. Well-made fonts include information for how differing amounts of space should be inserted between separate character pairs. This kerning space is inserted automatically but in rare circumstances you may wish to turn it off.

As briefly mentioned previously at the end of Section 16.2 on page 34, the Uppercase option will add a small amount of tracking between uppercase letters, seen in Example 30, which uses the Romande fonts⁸ (thanks to Clea F. Rees for the suggestion). The Uppercase option acts separately to the regular kerning controlled by the On/Off options.

16.6 Character width

Many Asian fonts are equipped with variously spaced characters for shoe-horning into their generally monospaced text. These are accessed through the CharacterWidth feature.

Japanese alphabetic glyphs (in Hiragana or Katakana) may be typeset proportionally, to better fit horizontal measures, or monospaced, to fit into the rigid grid imposed by ideographic typesetting. In this latter case, there are also half-width forms for squeezing more kana glyphs (which are less complex than the kanji they are amongst) into a given block of space. The same features are given to roman letters in Japanese fonts, for typesetting foreign words in the same style as the surrounding text.

The same situation occurs with numbers, which are provided in increasingly illegible compressed forms seen in Example 32.

Table 11: Options for the OpenType font feature 'Diacritics'.

Feature	Option	Tag	
Diacritics =	MarkToBase MarkToMark AboveBase BelowBase	mkmk abvm	†
	ResetAll	2 - WIII	_

[†] These feature options can be disabled with . .Off variants, and reset to default state (neither explicitly on nor off) with . .Reset.

Table 12: Options for the OpenType font feature 'Kerning'.

Feature	Option	Tag
Kerning =	On Off Reset	+kern -kern
	Uppercase ResetAll	cpsp †

[†] These feature options can be disabled with . .Off variants, and reset to default state (neither explicitly on nor off) with . .Reset.

Example 30: Adding extra kerning for uppercase letters. (The difference is usually very small.)

UPPERCASE EXAMPLE UPPERCASE EXAMPLE

\fontspec{Romande ADF Std Bold}
UPPERCASE EXAMPLE \\
\addfontfeature{Kerning=Uppercase}
UPPERCASE EXAMPLE

Table 13: Options for the OpenType font feature 'CharacterWidth'.

Feature	Option	Tag
CharacterWidth =	Proportional	pwid †
	Full	fwid †
	Half	hwid †
	Third	twid †
	Quarter	qwid †
	AlternateProportional	palt †
	AlternateHalf	halt †
	ResetAll	

 $[\]dagger$ These feature options can be disabled with . . Off variants, and reset to default state (neither explicitly on nor off) with . . Reset.

\def\test{\makebox[2cm][1]{\texta}%
 \makebox[2.5cm][1]{\textb}%
 \makebox[2.5cm][1]{abcdef}}

\fontspec{Hiragino Mincho Pro}

 $$$ {\addfontfeature{CharacterWidth=Full} \test} \\ {\addfontfeature{CharacterWidth=Half} \test}$

ようこそ ワカヨタレソ abcdef ようこそ ワカヨタレソ abcdef ようこそ ワカヨタレソ abcdef

Example 32: Numbers can be compressed significantly. \[\text{fontspec}[Renderer=AAT] \text{Hiragino Mincho Pro} \\ \text{\addfontfeature}{CharacterWidth=Full} \\ \text{---12321---}\\ \text{\addfontfeature}{CharacterWidth=Half} \\ \text{---1234554321---}\\ \text{-123455787654321-} \\ \text{\addfontfeature}{CharacterWidth=Third} \\ \text{---123456787654321---}\\ \text{\addfontfeature}{CharacterWidth=Quarter} \\ \text{---12345678900987654321---}\\ \text{---12345678900987654321---}\\ \end{addfontfeature} \]

Table 14: Options for the OpenType font feature 'CJKShape'.

Feature	Option	Tag
CJKShape =	Traditional	trad
	Simplified	smpl
	JIS1978	jp78
	JIS1983	jp83
	JIS1990	jp90
	Expert	expt
	NLC	nlck

 $[\]dagger$ These feature options can be disabled with . . Off variants, and reset to default state (neither explicitly on nor off) with . . Reset.

16.6.1 CJK shape

There have been many standards for how CJK ideographic glyphs are 'supposed' to look. Some fonts will contain many alternate glyphs available in order to be able to display these gylphs correctly in whichever form is appropriate. Both AAT and OpenType fonts support the following CJKShape options: Traditional, Simplified, JIS1978, JIS1983, JIS1990, and Expert. OpenType also supports the NLC option.

16.7 Vertical typesetting

OpenType provides a plethora of features for accommodating the varieties of possibilities needed for vertical typesetting (CJK and others). No capabilities for achieving such vertical typesetting are provided by fontspec, however; please get in touch if there are improvements that could be made.

16.8 Numeric features

16.8.1 Stylistic Set variations — ssnn

This feature selects a 'Stylistic Set' variation, which usually corresponds to an alternate glyph style for a range of characters (usually an alphabet or subset thereof). This feature is specified numerically. These correspond to OpenType features sso1, sso2, etc.

Two demonstrations from the Junicode font⁹ are shown in Example 34 and Example 35; thanks to Adam Buchbinder for the suggestion.

Multiple stylistic sets may be selected simultaneously by writing, e.g., StylisticSet={1,2,3}.

The StylisticSet feature is a synonym of the Variant feature for AAT fonts. See Section 23 on page 65 for a way to assign names to stylistic sets, which should be done on a per-font basis.

16.8.2 Character Variants — cvNN

Similar to the 'Stylistic Sets' above, 'Character Variations' are selected numerically to adjust the output of (usually) a single character for the particular font. These correspond to the OpenType features cvo1 to cv99.

 ${\it Example 33: Different standards for CJK ideograph presentation.}$

	\fontspec{Hiragino Mincho Pro}
医喉炎 牡头红	{\addfontfeature{CJKShape=Traditional}
唖噛躯 妍并訝	\text } \\
唖噛躯 姸幷訝	{\addfontfeature{CJKShape=NLC}
型物 ががた	\text } \\
啞嚙騙 妍并訝	{\addfontfeature{CJKShape=Expert}
""四河" 外了下的	\text }

⁸ http://arkandis.tuxfamily.org/adffonts.html

⁹http://junicode.sf.net

Table 15: Options for the OpenType font feature 'Vertical'.

Feature	Option	Tag	
Vertical =	RotatedGlyphs	vrt2	+
	AlternatesForRotation	vrtr	+
	Alternates	vert	+
	KanaAlternates	vkna	+
	Kerning	vkrn	+
	AlternateMetrics	valt	+
	HalfMetrics	vhal	†
	ProportionalMetrics	vpal	†
	ResetAll		

 $[\]dagger$ These feature options can be disabled with . . Off variants, and reset to default state (neither explicitly on nor off) with . . Reset.

Example 34: Insular letterforms, as used in medieval Northern Europe, for the Junicode font accessed with the StylisticSet feature.

Insular forms.	\fontspec{Junicode} Insular forms. \\
Inrulap ropmr.	\addfontfeature{StylisticSet=2} Insular forms. \\

Example 35: Enlarged minuscules (capital letters remain unchanged) for the Junicode font, accessed with the StylisticSet feature.

\fontspec{Junicode} ENLARGED Minuscules. \\ \addfontfeature{StylisticSet=6}
ENLARGED Minuscules. \\

For each character that can be varied, it is possible to select among possible options for that particular glyph. For example, in Example 36 a variety of glyphs for the character 'v' are selected, in which 5 corresponds to the character 'v' for this font feature, and the trailing : $\langle n \rangle$ corresponds to which variety to choose. Georg Duffner's open source Garamond revival font¹⁰ is used in this example. Character variants are specifically designed not to conflict with each other, so you can enable them individually per character as shown in Example 37. (Unlike stylistic alternates, say.)

Note that the indexing starts from zero.

16.8.3 Alternates — salt

The Alternate feature, alias StylisticAlternates, is used to access alternate font glyphs when variations exist in the font, such as in Example 38. It uses a numerical selection, starting from zero, that will be different for each font. Note that the Style=Alternate option is equivalent to Alternate=o to access the default case.

Note that the indexing starts from zero. With the LuaTeX engine, Alternate=Random selects a random alternate.

See Section 23 on page 65 for a way to assign names to alternates if desired.

16.8.4 Annotation — nalt

Some fonts are equipped with an extensive range of numbers and numerals in different forms. These are accessed with the Annotation feature (OpenType feature nalt), selected numerically as shown in Example 39. Note that the indexing starts from zero.

16.8.5 Ornament — ornm

Ornaments are selected with the Ornament feature (OpenType feature ornm), selected numerically such as for the Annotation feature. If you know of an Open Source font that supports this feature, let me know and I'll add an example.

16.9 OpenType scripts and languages

Fonts that include glyphs for various scripts and languages may contain different font features for the different character sets and languages they support, and different font features may behave differently depending on the script or language chosen. When multilingual fonts are used, it is important to select which language they are being used for, and more importantly what script is being used.

The 'script' refers to the alphabet in use; for example, both English and French use the Latin script. Similarly, the Arabic script can be used to write in both the Arabic and Persian languages.

The Script and Language features are used to designate this information. The possible options are tabulated in Table 16 on page 51 and Table 17 on page 52, respectively. When a script or language is requested that is not supported by the current font, a warning is printed in the console output.

Because these font features can change which features are able to be selected for the font, they are automatically selected by fontspec before all others and, if X₃T_EX is

 $^{^{\}mathbf{10}} \mathtt{http://www.georgduffner.at/ebgaramond/}$

Example 36: The CharacterVariant feature showing off Georg Duffner's open source Garamond revival font.

 ${\tt Example~37:~The~CharacterVariant~feature~selecting~multiple~variants~simultaneously}.$

```
& violet

& violet

& violet

& violet

\fontspec{EB Garamond 12 Italic}

\fontspec{EB Garamond 12 Italic}

\& violet \\
fontspec{EB Garamond 12 Italic}

\fontspec{EB Garamond 12 Italic}

\fontspec{EB Garamond 12 Italic}

\fontspec{EB Garamond 12 Italic}

\& violet \\
fontspec{EB Garamond 12 Italic}
```

Example 39: Annotation forms for OpenType fonts.

```
123456789
(1) (2) (3) (4) (5) (6) (7) (8) (9)
(1 (2 (3 (4 (5 (6 (7 (8 (9
1) 2) 3) 4) 5) 6) 7) 8) 9)
1 2 3 4 5 6 7 8 9
0 2 8 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
                          \fontspec{Hiragino Maru Gothic Pro}
123456789
                           1 2 3 4 5 6 7 8 9
123456789
                          \def\x#1{\\{\addfontfeature{Annotation=#1}}
1 2 3 4 5 6 7 8 9
                                  123456789}}
1. 2. 3. 4. 5. 6. 7. 8. 9.
                          \xo\x1\x2\x3\x4\x5\x6\x7\x7\x8\x9
```

being used, will specifically select the OpenType renderer for this font, as described in Section 21.2 on page 59.

See Section 24 on page 66 for methods to create new Script or Language options if required.

16.9.1 Script and Language examples

In the examples shown in Example 40, the Code2000 font¹¹ is used to typeset various input texts with and without the OpenType Script applied for various alphabets. The text is only rendered correctly in the second case; many examples of incorrect diacritic spacing as well as a lack of contextual ligatures and rearrangement can be seen. Thanks to Jonathan Kew, Yves Codet and Gildas Hamel for their contributions towards these examples.

[&]quot;1http://www.code2000.net/

Example 40: An example of various Scripts and Languages.

العربي العربي हिन्दी हिन्दी ल८ एल ए

મર્યાદા-સૂયક નવિદન ર્મયાદા-સૂયક નિવેદન നമ്മുടെ പാരബര്യ നമ്മുടെ പാരബര്യ ਆਦ ਸਿਚੂ ਜੁਗਾਦ ਸਿਚੂ ਆਦਿ ਸਚੂ ਜੁਗਾਦਿ ਸਚੂ தமிழ் தடேி தமிழ் தேடி

רִדְתַּה רִבְּתה

cấp số mỗi cấp số mỗi

\testfeature{Script=Arabic}{\arabictext}

\testfeature{Script=Devanagari}{\devanagaritext}

\testfeature{Script=Bengali}{\bengalitext} \testfeature{Script=Gujarati}{\gujaratitext} \testfeature{Script=Malayalam}{\malayalamtext} \testfeature{Script=Gurmukhi}{\gurmukhitext} \testfeature{Script=Tamil}{\tamiltext}

\testfeature{Script=Hebrew}{\hebrewtext}

\def\examplefont{Doulos SIL}

\testfeature{Language=Vietnamese}{\vietnamesetext}

Table 16: Defined Scripts for OpenType fonts. Aliased names are shown in adjacent positions marked with red pilcrows (\mathfrak{q}).

Arabic Ethiopic Armenian Georgian Balinese Glagolitic Bengali Gothic Bopomofo Greek Braille Gujarati Buginese Gurmukhi Buhid Byzantine Music Canadian Syllabics Cherokee **q**CJK **q**CJK Ideographic Coptic Cypriot Syllabary Cyrillic

Default

Deseret

Devanagari

Hangul Jamo Hangul Hanunoo Hebrew **q**Hiragana and Katakana Javanese Kannada Kharosthi Khmer Lao Latin

Limbu Linear B Malayalam **q**Math **q** Maths Mongolian Musical Symbols Myanmar N′ko ${\sf Ogham}$ Old Italic Old Persian Cuneiform Oriya Osmanya

Phags-pa Phoenician Runic Shavian Sinhala

Sumero-Akkadian Cuneiform Syloti Nagri Syriac Tagalog Tagbanwa Tai Le Tai Lu Tamil Telugu Thaana Thai Tibetan Tifinagh Ugaritic Cuneiform

Table 17: Defined Languages for OpenType fonts. Aliased names are shown in adjacent positions marked with red pilcrows (q).

Abaza	Default	Igbo	Koryak	Norway House Cree	Serer
Abkhazian	Dogri	ljo	Ladin	Nisi	South Slavey
Adyghe	Divehi	Ilokano	Lahuli	Niuean	Southern Sami
Afrikaans	Djerma	Indonesian	Lak	Nkole	Suri
Afar	Dangme	Ingush	Lambani	N'ko	Svan
Agaw	Dinka	Inuktitut	Lao	Dutch	Swedish
Altai	Dungan	Irish	Latin	Nogai	Swadaya Aramaic
Amharic	Dzongkha	Irish Traditional	Laz	Norwegian	Swahili
Arabic	Ebira	Icelandic	L-Cree	Northern Sami	Swazi
Aari	Eastern Cree	Inari Sami	Ladakhi	Northern Tai	Sutu
Arakanese	Edo	Italian	Lezgi	Esperanto	Syriac
Assamese	Efik	Hebrew	Lingala	Nynorsk	Tabasaran
Athapaskan	Greek	lavanese	Low Mari	Oji-Cree	Tajiki
Avar	English	Yiddish	Limbu	Ojibway	Tamil
Awadhi	Erzya	Japanese	Lomwe	Oriya	Tatar
Aymara	Spanish	Judezmo	Lower Sorbian	Oromo	TH-Cree
Azeri	Estonian	Jula	Lule Sami	Ossetian	Telugu
Badaga	Basque	Kabardian	Lithuanian	Palestinian Aramaic	Tongan
Baghelkhandi	Evenki	Kachchi	Luba	Pali	Tigre
Balkar	Even	Kalenjin	Luganda	Punjabi	Tigrinya
Baule	Ewe	Kannada	Luhya	Palpa	Thai
Berber	French Antillean	Karachay	Luo	Pashto	Tahitian
Bench	⊄ Farsi	Georgian	Latvian	Polytonic Greek	Tibetan
Bible Cree	⊈ Parsi	Kazakh	Majang	Pilipino	Turkmen
Belarussian	⊄ Persian	Kebena	Makua	Palaung	Temne
Bemba	Finnish	Khutsuri Georgian	Malayalam	Polish	Tswana
	Fijian	Khakass	Traditional	Provencal	Tundra Nenets
Bengali Bulgarian	Flemish		Mansi		
Bulgarian Bhili	Forest Nenets	Khanty-Kazim Khmer	Marathi	Portuguese Chin	Tonga Todo
	Fon	Khanty-Shurishkar	Marwari	Rajasthani	Turkish
Bhojpuri Bikol	Faroese	Khanty-Vakhi	Mbundu	R-Cree	Tsonga
Bilen	French	Khowar	Manchu		Turoyo Aramaic
Blackfoot	Frisian	Kikuyu	Moose Cree	Russian Buriat Riang	Tulu
Balochi	Friulian	,	Mende	Rhaeto-Romanic	Tuvin
Balante	Futa	Kirghiz Kisii	Me'en	Romanian	Twi
Balti	Fulani	Kokni	Mizo	Romany	Udmurt
Bambara	Ga	Kalmyk	Macedonian	Rusyn	Ukrainian
Bamileke	Gaelic	Kamba	Male	Ruanda	Urdu
Breton	Gagauz	Kumaoni		Russian	Upper Sorbian
Brahui	Galician	Kumaoni Komo	Malagasy Malinke	Sadri	1.1
	Garshuni	Komso		Sanskrit	Uyghur Uzbek
Braj Bhasha		Kanuri	Malayalam		
Burmese Bashkir	Garhwali Ge'ez		Reformed	Santali	Venda
		Kodagu Karaan Old Hangul	Malay	Sayisi	Vietnamese
Beti Catalan	Gilyak Gumuz	Korean Old Hangul	Mandinka	Sekota	Wa Waadi
		Konkani	Mongolian	Selkup	Wagdi
Cebuano Chechen	Gondi Greenlandic	Kikongo	Manipuri Maninka	Sango	West-Cree Welsh
		Komi-Permyak		Shan Sibe	Wolof
Chaha Gurage	Garo	Korean	Manx Gaelic		
Chattisgarhi	Guarani	Komi-Zyrian	Moksha Moldavian	Sidamo	Tai Lue
Chichewa	Gujarati	Kpelle		Silte Gurage	Xhosa
Chukchi	Haitian	Krio	Mon	Skolt Sami	Yakut
Chipewyan	Halam	Karakalpak	Moroccan	Slovak	Yoruba Y-Cree
Cherokee	Harauti	Karelian	Maori	Slavey	
Chuvash	Hausa	Karaim	Maithili	Slovenian	Yi Classic
Comorian	Hawaiin	Karen	Maltese	Somali	Yi Modern
Coptic	Hammer-Banna	Koorete	Mundari	Samoan	Chinese Hong Kong
Cree	Hiligaynon	Kashmiri	Naga-Assamese	Sena	Chinese Phonetic
Carrier	Hindi	Khasi	Nanai	Sindhi	Chinese Simplified
Crimean Tatar	High Mari	Kildin Sami	Naskapi	Sinhalese	Chinese Traditional
Church Slavonic	Hindko	Kui	N-Cree	Soninke	Zande
Czech	Ho	Kulvi	Ndebele	Sodo Gurage	Zulu
Danish	Harari	Kumyk	Ndonga	Sotho	
Dargwa	Croatian	Kurdish	Nepali	Albanian	
Woods Cree	Hungarian	Kurukh	Newari	Serbian	
German	Armenian	Kuy	Nagari	Saraiki	

Part V

Commands for accents and symbols ('encodings')

The functionality described in this section is experimental.

In the pre-Unicode era, significant work was required by LTEX to ensure that input characters in the source could be interpreted correctly depending on file encoding, and that glyphs in the output were selected correctly depending on the font encoding. With Unicode, we have the luxury of a single file and font encoding that is used for both input and output.

While this may provide some illusion that we could get away simply with typing Unicode text and receive correct output, this is not always the case. For a start, hyphenation in particular is language-specific, so tags should be used when switch between languages in a document. The babel and polyglossia packages both provide features for this.

Multilingual documents will often use different fonts for different languages, not just for style, but for the more pragmatic reason that fonts do not all contain the same glyphs. (In fact, only test fonts such as Code2000 provide anywhere near the full Unicode coverage.) Indeed, certain fonts may be perfect for a certain application but miss a handful of necessary diacritics or accented letters. In these cases, fontspec can leverage the font encoding technology built into MEX2 to provide on a per-font basis either provide fallback options or error messages when a desired accent or symbol is not available. However, at present these features can only be provided for input using MEX commands rather than Unicode input; for example, typing \`e instead of \ealso or \textcopyright instead of \ealso in the source file.

The most widely-used encoding in \LaTeX 2 $_{\mathcal{E}}$ was T1 with companion 'TS1' symbols provided by the textcomp package. These encodings provided glyphs to typeset text in a variety of western European languages. As with most legacy \LaTeX 2 $_{\mathcal{E}}$ input methods, accents and symbols were input using encoding-dependent commands such as \`e as described above. As of 2017, in \LaTeX 2 $_{\mathcal{E}}$ on చ and LuaTeX, the default encoding is TU, which uses Unicode for input and output. The TU encoding provides appropriate encoding-dependent definitions for input commands to match the coverage of the T1+TS1 encodings. Wider coverage is not provided by default since (a) each font will provide different glyph coverage, and (b) it is expected that most users will be writing with direct Unicode input.

For those users who do need finer-grained control, fontspec provides an interface for a more extensible system.

17 A new Unicode-based encoding from scratch

Let's say you need to provide support for a document originally written with fonts in the OT2 encoding, which contains encoding-dependent commands for Cyrillic letters. An example from the OT2 encoding definition file (ot2enc.def) reads:

```
57 \DeclareTextSymbol{\CYRIE}{0T2}{5}
58 \DeclareTextSymbol{\CYRDJE}{0T2}{6}
59 \DeclareTextSymbol{\CYRTSHE}{0T2}{7}
60 \DeclareTextSymbol{\cyrnje}{0T2}{8}
61 \DeclareTextSymbol{\cyrlje}{0T2}{9}
62 \DeclareTextSymbol{\cyrdzhe}{0T2}{10}
```

To recreate this encoding in a form suitable for fontspec, create a new file named, say, fontrange-cyr.def and populate it with

```
\DeclareTextSymbol{\CYRIE} {\LastDeclaredEncoding}{\"0404} \DeclareTextSymbol{\CYRDJE} {\LastDeclaredEncoding}{\"0402} \DeclareTextSymbol{\CYRTSHE}{\LastDeclaredEncoding}{\"040B} \DeclareTextSymbol{\cyrnje} {\LastDeclaredEncoding}{\"045A} \DeclareTextSymbol{\cyrlje} {\LastDeclaredEncoding}{\"0459} \DeclareTextSymbol{\cyrdzhe}{\LastDeclaredEncoding}{\"045F}
```

The numbers "0404, "0402, ..., are the Unicode slots (in hexadecimal) of each glyph respectively. The fontspec package provides a number of shorthands to simplify this style of input; in this case, you could also write

```
\EncodingSymbol{\CYRIE}{"0404}
...
```

To use this encoding in a fontspec font, you would first add this to your preamble:

```
\DeclareUnicodeEncoding{unicyr}{
  \input{fontrange-cyr.def}
}
```

Then follow it up with a font loading call such as

```
\setmainfont{...}[NFSSEncoding=unicyr]
```

The first argument unicyr is the name of the 'encoding' to use in the font family. (There's nothing special about the name chosen but it must be unique.) The second argument to \DeclareUnicodeEncoding also allows adjustments to be made for perfont changes. We'll cover this use case in the next section.

18 Adjusting a pre-existing encoding

There are three reasons to adjust a pre-existing encoding: to add, to remove, and to redefine some symbols, letters, and/or accents.

When adding symbols, etc., simply write

```
\DeclareUnicodeEncoding{unicyr}{
  \input{tuenc.def}
  \input{fontrange-cyr.def}
  \EncodingSymbol{\textruble}{"20BD}}
```

Of course if you consistently add a number of symbols to an encoding it would be a good idea to create a new fontrange-XX.def file to suit your needs.

When removing symbols, use the $\UndeclareSymbol{\langle cmd \rangle}$ command. For example, if you a loading a font that you know is missing, say, the interrobang (not that unusual a situation), you might write:

```
\DeclareUnicodeEncoding{nobang}{
  \input{tuenc.def}
  \UndeclareSymbol\textinterrobang
}
```

Provided that you use the command \textinterrobang to typeset this symbol, it will appear in fonts with the default encoding, while in any font loaded with the nobang encoding an attempt to access the symbol will either use the default fallback definition or return an error, depending on the symbol being undeclared.

The third use case is to redefine a symbol or accent. The most common use case in this scenario is to adjust a specific accent command to either fine-tune its placement or to 'fake' it entirely. For example, the underdot diacritic is used in typeset Sanskrit, but it is not necessarily included as an accent symbol is all fonts. By default the underdot is defined in TU as:

```
\EncodingAccent{\d}{"0323}
```

For fonts with a missing (or poorly-spaced) "0323 accent glyph, the 'traditional' TEX fake accent construction could be used instead:

```
\DeclareUnicodeEncoding{fakeacc}{
  \input{tuenc.def}
  \EncodingCommand{\d}[1]{%
    \hmode@bgroup
    \o@lign{\relax#1\crcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}%
    \egroup
  }
}
```

This would be set up in a document as such:

```
\newfontfamily\sanskitfont{CharisSIL}
\newfontfamily\titlefont{Posterama}[NFSSEncoding=fakeacc]
```

Then later in the document, no additional work is needed:

```
...{\titlefont kalita\d m}... % <- uses fake accent ...{\sanskitfont kalita\d m}... % <- uses real accent
```

To reiterate from above, typing this input with Unicode text ('kalitam') will bypass this encoding mechanism and you will receive only what is contained literally within the font.

19 Summary of commands

The $\text{MEX}\ 2\varepsilon$ kernel provides the following font encoding commands suitable for Unicode encodings:

See fntguide.pdf for full documentation of these. As shown above, the following shorthands a provided by fontspec to simplify the process of defining Unicode font range encodings:

```
\label{lem:command} $$ \operatorname{Command}_{\langle code\rangle} $$ \operatorname{CodingAccent}_{\langle command\rangle}_{\langle code\rangle} $$ \operatorname{CodingSymbol}_{\langle command\rangle}_{\langle code\rangle} $$ \operatorname{CodingComposite}_{\langle command\rangle}_{\langle code\rangle} $$ \operatorname{Command}_{\langle command\rangle}_{\langle code\rangle} $$ \operatorname{Command}_{\langle command\rangle}_{\langle code\rangle}_{\langle code\rangle}_{\langle code\rangle}_{\langle command\rangle}_{\langle command\rangle}_{\langle command\rangle}_{\langle code\rangle}_{\langle code\rangle}_{\langle command\rangle}_{\langle command\rangle}_{\langle
```

Despite its name, \UndeclareSymbol can be used for commands defined by all three of \EncodingCommand, \EncodingAccent, and \EncodingSymbol.

Part VI

LuaT_EX-only font features

20 Custom font features

Pre-2016, it was possible to load an OpenType font feature file to define new OpenType features for a selected font. This facility was particularly useful to implement custom substitutions, for example. As of TeXLive 2016, LuaTeX/luaotfload no longer supports this feature, but provides its own internal mechanisms for an equivalent interface.

Any documents using 'feature file' options will need to transition to the new interface. Figure ${\tt I}$ shows an example. Please refer to the LuaTeX/luaotfload documentation for more details.

Figure 1: An example of custom font features.

```
\documentclass{article}
\usepackage{fontspec}
\directlua{
   fonts.handlers.otf.addfeature {
       name = "oneb",
        {
            type = "substitution",
            data = {
               ["1"] = "one.sso1",
            },
        },
        "feature oneb for vollkorn font",
   }
}
\setmainfont{Vollkorn-Regular.otf}[RawFeature=+oneb]
\begin{document}
1234567890
\end{document}
```

Part VII

Fonts and features with X_HT_EX

21 X₃T_EX-only font features

The features described here are available for any font selected by fontspec.

21.1 Mapping

Mapping enables a X¬TEX text-mapping scheme, shown in Example 41.

Using the tex-text mapping is also equivalent to writing Ligatures=TeX. The use of the latter syntax is recommended for better compatibility with LuaTeX documents.

21.2 Different font technologies: AAT and OpenType

X₃T_EX supports two rendering technologies for typesetting, selected with the Renderer font feature. The first, AAT, is that provided (only) by Mac OS X itself. The second, OpenType, is an open source OpenType interpreter. ¹² It provides greater support for OpenType features, notably contextual arrangement, over AAT.

In general, this feature will not need to be explicitly called: for OpenType fonts, the OpenType renderer is used automatically, and for AAT fonts, AAT is chosen by default. Some fonts, however, will contain font tables for both rendering technologies, such as the Hiragino Japanese fonts distributed with Mac OS X, and in these cases the choice may be required.

Among some other font features only available through a specific renderer, Open—Type provides for the Script and Language features, which allow different font behaviour for different alphabets and languages; see Section 16.9 on page 48 for the description of these features. Because these font features can change which features are able to be selected for the font instance, they are selected by fontspec before all others and will automatically and without warning select the OpenType renderer.

21.3 Optical font sizes

Multiple Master fonts are parameterised over orthogonal font axes, allowing continuous selection along such features as weight, width, and optical size (see ?? on page ?? for further details). Whereas an OpenType font will have only a few separate optical sizes, a Multiple Master font's optical size can be specified over a continuous range.

 $^{^{12}}$ v2.4: This was called 'ICU' in previous versions of X $_{\overline{1}}$ TeX and fontspec. Backwards compatibility is preserved.

Example 41: X <u>J</u> T <u>E</u> X's Mapping feature.		
"¡A small amount of—text!"	\fontspec{Cochin}[Mapping=tex-text] ``!`A small amount oftext!''	

Unfortunately, this flexibility makes it harder to create an automatic interface through Lagrange to a Multiple Master font must always be specified explicitly.

```
\fontspec{Minion MM Roman}[OpticalSize=11]
MM optical size test  \\
\fontspec{Minion MM Roman}[OpticalSize=47]
MM optical size test  \\
\fontspec{Minion MM Roman}[OpticalSize=71]
MM optical size test  \\
```

22 Mac OS X's AAT fonts

Warning! X₃T_EX's implementation on Mac OS X is currently in a state of flux and the information contained below may well be wrong from 2013 onwards. There is a good chance that the features described in this section will not be available any more as X₃T_EX's completes its transition to a cross-platform-only application.

Mac OS X's font technology began life before the ubiquitous-OpenType era and revolved around the Apple-invented 'AAT' font format. This format had some advantages (and other disadvantages) but it never became widely popular in the font world.

Nonetheless, this is the font format that was first supported by $X_{\exists}T_{\exists}X$ (due to its pedigree on Mac OS X in the first place) and was the first font format supported by fontspec. A number of fonts distributed with Mac OS X are still in the AAT format, such as 'Skia'.

22.1 Ligatures

Ligatures refer to the replacement of two separate characters with a specially drawn glyph for functional or æsthetic reasons. For AAT fonts, you may choose from any combination of Required, Common, Rare (or Discretionary), Logos, Rebus, Diphthong, Squared, AbbrevSquared, and Icelandic.

Some other Apple AAT fonts have those 'Rare' ligatures contained in the Icelandic feature. Notice also that the old TeX trick of splitting up a ligature with an empty brace pair does not work in XaTeX; you must use a opt kern or \hbox (e.g., \null) to split the characters up if you do not want a ligature to be performed (the usual examples for when this might be desired are words like 'shelffull').

22.2 Letters

The Letters feature specifies how the letters in the current font will look. For AAT fonts, you may choose from Normal, Uppercase, Lowercase, SmallCaps, and InitialCaps.

22.3 Numbers

The Numbers feature defines how numbers will look in the selected font. For AAT fonts, they may be a combination of Lining or OldStyle and Proportional or Monospaced

(the latter is good for tabular material). The synonyms Uppercase and Lowercase are equivalent to Lining and OldStyle, respectively. The differences have been shown previously in Section 10 on page 22.

22.4 Contextuals

This feature refers to glyph substitution that vary by their position; things like contextual swashes are implemented here. The options for AAT fonts are WordInitial, WordFinal (Example 42), LineInitial, LineFinal, and Inner (Example 43, also called 'non-final' sometimes). As non-exclusive selectors, like the ligatures, you can turn them off by prefixing their name with No.

22.5 Vertical position

The VerticalPosition feature is used to access things like subscript (Inferior) and superscript (Superior) numbers and letters (and a small amount of punctuation, sometimes). The Ordinal option is (supposed to be) contextually sensitive to only raise characters that appear directly after a number. These are shown in Example 44.

The realscripts package (also loaded by xltxtra) redefines the \textsubscript and \textsuperscript commands to use the above font features, including for use in footnote labels.

22.6 Fractions

Many fonts come with the capability to typeset various forms of fractional material. This is accessed in fontspec with the Fractions feature, which may be turned On or Off in both AAT and OpenType fonts.

In AAT fonts, the 'fraction slash' or solidus character, is to be used to create fractions. When Fractions are turned On, then only pre-drawn fractions will be used. See Example 45.

Using the Diagonal option (AAT only), the font will attempt to create the fraction from superscript and subscript characters.

Some (Asian fonts predominantly) also provide for the Alternate feature shown in Example 46.

22.7 Variants

The Variant feature takes a single numerical input for choosing different alphabetic shapes. Don't mind my fancy Example 47:) I'm just looping through the nine (!) variants of Zapfino.

Example 42: Contextual glyph for the beginnings and ends of words.	
	\newfontface\fancy{Hoefler Text Italic}[%
where is all the vegemite	<pre>Contextuals={WordInitial,WordFinal}]</pre>
	\fancy where is all the vegemite

Example 43: A contextual feature for the 'long s' can be convenient as the character does not need to be marked up explicitly.

'Inner' fwashes can fometimes contain the archaic long s. \fontspec{Hoefler Text}[Contextuals=Inner] \Inner' swashes can \emph{sometimes} \\ contain the archaic long~s.

Example 44: Vertical position for AAT fonts. \[\begin{align*} \fontspec{Skia} \\ Normal \\ fontspec{Skia} \[VerticalPosition=Superior \\ Superior \\ fontspec{Skia} \[VerticalPosition=Inferior \\ Inferior \\ \\ Normal \superior \\ inferior \\ 1st 2nd 3rd 4th 0th 8abcde \end{align*} \] Normal superior inferior \\ \fontspec{Skia} \[VerticalPosition=Ordinal \] \\ 1st 2nd 3rd 4th oth 8abcde \end{align*}

Example 45: Fractions in AAT fonts. The ^^^2044 glyph is the 'fraction slash' that may be typed in Mac OS X with OPT+SHIFT+1; not shown literally here due to font contraints.

Example 46: Alternate design of pre-composed fractions.

Example 47: Nine variants of Zapfino.



\newcounter{var}
\whiledo{\value{var}<g}{%
 \edef\1{%
 \noexpand\fontspec[Variant=\thevar,
 Color=oogg\thevar\thevar]{Zapfino}}\1%
 \makebox[0.75\width]{d}%
 \stepcounter{var}}
\hspace*{2cm}</pre>

See Section 23 on page 65 for a way to assign names to variants, which should be done on a per-font basis.

22.8 Alternates

Selection of Alternates again must be done numerically; see Example 48. See Section 23 on page 65 for a way to assign names to alternates, which should be done on a per-font basis.

22.9 Style

The options of the Style feature are defined in AAT as one of the following: Display, Engraved, IlluminatedCaps, Italic, Ruby, ¹³ TallCaps, or TitlingCaps.

Typical examples for these features are shown in Section 16.3.

22.10 CJK shape

There have been many standards for how CJK ideographic glyphs are 'supposed' to look. Some fonts will contain many alternate glyphs in order to be able to display these gylphs correctly in whichever form is appropriate. Both AAT and OpenType fonts support the following CJKShape options: Traditional, Simplified, JIS1978, JIS1983, JIS1990, and Expert. OpenType also supports the NLC option.

Example 48: Alternate shape selection must be numerical.

Sphinx Of Black Quartz, Judge Mr Vow Sphinx Of Black Quartz, Judge Mr Vow \fontspec{Hoefler Text Italic}[Alternate=0]
Sphinx Of Black Quartz, {\scshape Judge My Vow} \\
\fontspec{Hoefler Text Italic}[Alternate=1]
Sphinx Of Black Quartz, {\scshape Judge My Vow}

¹³ 'Ruby' refers to a small optical size, used in Japanese typography for annotations.

22.11 Character width

See Section 16.6 on page 43 for relevant examples; the features are the same between OpenType and AAT fonts. AAT also allows CharacterWidth=Default to return to the original font settings.

22.12 Vertical typesetting

X₃T_EX provides for vertical typesetting simply with the ability to rotate the individual glyphs as a font is used for typesetting, as shown in Example 49.

No actual provision is made for typesetting top-to-bottom languages; for an example of how to do this, see the vertical Chinese example provided in the X¬TEX documentation.

22.13 Diacritics

Diacritics are marks, such as the acute accent or the tilde, applied to letters; they usually indicate a change in pronunciation. In Arabic scripts, diacritics are used to indicate vowels. You may either choose to Show, Hide or Decompose them in AAT fonts. The Hide option is for scripts such as Arabic which may be displayed either with or without vowel markings. E.g., \fontspec[Diacritics=Hide] { . . . }

Some older fonts distributed with Mac OS X included '0/' etc. as shorthand for writing 'Ø' under the label of the Diacritics feature. If you come across such fonts, you'll want to turn this feature off (imagine typing hello/goodbye and getting 'helløgoodbye' instead!) by decomposing the two characters in the diacritic into the ones you actually want. I recommend using the proper LateX input conventions for obtaining such characters instead.

22.14 Annotation

Various Asian fonts are equipped with a more extensive range of numbers and numerals in different forms. These are accessed through the Annotation feature with the following options: Off, Box, RoundedBox, Circle, BlackCircle, Parenthesis, Period, RomanNumerals, Diamond, BlackSquare, BlackRoundSquare, and Double-Circle.

Example 49: Ve	rtical typesettin	g
----------------	-------------------	---

共産主義者は

共産主義者

\fontspec{Hiragino Mincho Pro}
\verttext

Part VIII

Customisation and programming interface

This is the beginning of some work to provide some hooks that use fontspec for various macro programming purposes.

23 Defining new features

This package cannot hope to contain every possible font feature. Three commands are provided for selecting font features that are not provided for out of the box. If you are using them a lot, chances are I've left something out, so please let me know.

\newAATfeature

\newopentypefeature

New AAT features may be created with this command:

 $\verb|\newAATfeature|{\langle feature \rangle}|{\langle option \rangle}|{\langle feature\ code \rangle}|{\langle selector\ code \rangle}|$

Use the X₃T_EX file AAT-info.tex to obtain the code numbers. See Example 50.

New OpenType features may be created with this command:

 $\new open type feature \{\langle feature \rangle\} \{\langle option \rangle\} \{\langle feature \ tag \rangle\}$

The synonym \newICUfeature is deprecated.

Here's what it would look like in practise:

\newopentypefeature{Style}{NoLocalForms}{-locl}

\newfontfeature

In case the above commands do not accommodate the desired font feature (perhaps a new X₁T_EX feature that fontspec hasn't been updated to support), a command is provided to pass arbitrary input into the font selection string:

```
\newfontfeature{\langle name \rangle} {\langle input string \rangle}
```

For example, Zapfino used to contain an AAT feature 'Avoid d-collisions'. To access it with this package, you could do some like the following:

```
\newfontfeature{AvoidD} {Special= Avoid d-collisions}
\newfontfeature{NoAvoidD}{Special=!Avoid d-collisions}
\fontspec{Zapfino}[AvoidD, Variant=1]
    sockdolager rubdown  \\
\fontspec{Zapfino}[NoAvoidD, Variant=1]
    sockdolager rubdown
```

The advantage to using the \newAATfeature and \newopentypefeature commands instead of \newfontfeature is that they check if the selected font actually

Example 50: Assigning new AAT features.

This is XeTeX by Jonathan Kew.

\newAATfeature{Alternate}{HoeflerSwash}{17}{1}
\fontspec{Hoefler Text Italic}[Alternate=HoeflerSwash]
This is XeTeX by Jonathan Kew.

contains the desired font feature at load time. By contrast, \newfontfeature will not give a warning for improper input.

24 Defining new scripts and languages

\newfontscript \newfontlanguage

While the scripts and languages listed in Table 16 and Table 17 are intended to be comprehensive, there may be some missing; alternatively, you might wish to use different names to access scripts/languages that are already listed. Adding scripts and languages can be performed with the \newfontscript and \newfontlanguage commands. For example,

```
\newfontscript{Arabic}{arab}
\newfontlanguage{Zulu}{ZUL}
```

The first argument is the fontspec name, the second the OpenType tag. The advantage to using these commands rather than \newfontfeature (see Section 23 on the preceding page) is the error-checking that is performed when the script or language is requested.

25 Going behind fontspec's back

Expert users may wish not to use fontspec's feature handling at all, while still taking advantage of its Lagrange font selection conveniences. The RawFeature font feature allows font feature selection using a literal feature selection string if you happen to have the OpenType feature tag memorised.

Multiple features can either be included in a single declaration:

[RawFeature=+smcp;+onum]

or with multiple declarations:

[RawFeature=+smcp, RawFeature=+onum]

26 Renaming existing features & options

\aliasfontfeature

If you don't like the name of a particular font feature, it may be aliased to another with the \alias fontfeature { $\langle existing\ name \rangle$ } { $\langle new\ name \rangle$ } command, such as shown in Example 52.

Spaces in feature (and option names, see below) are allowed. (You may have noticed this already in the lists of OpenType scripts and languages).

If you wish to change the name of a font feature option, it can be aliased to another

\aliasfontfeatureoption

Example 51: Using raw font features directly.

\[\fontspec\{texgyrepagella-regular.otf}\{ [RawFeature=+smcp] \] \] Pagella small caps

Example	52: Renaming font features.
Roman Letters And Swash	\aliasfontfeature{ItalicFeatures}{IF} \fontspec{Hoefler Text}[IF = {Alternate=1}] Roman Letters \itshape And Swash
Fxample 53	: Renaming font feature options.

\aliasfontfeature{VerticalPosition}{Vert Pos}
\aliasfontfeatureoption{VerticalPosition}{ScientificInferior}{Sci Inf}
\fontspec{LinLibertine_R.otf}[Vert Pos=Sci Inf]

rior: 12345
Scientific Inferior: 12345

Scientific Inferior: 12345

with the command \aliasfontfeatureoption{ $\langle fontfeature \rangle$ }{ $\langle existing\ name \rangle$ }{ $\langle new\ name \rangle$ }, such as shown in Example 53.

This example demonstrates an important point: when aliasing the feature options, the original feature name must be used when declaring to which feature the option belongs.

Only feature options that exist as sets of fixed strings may be altered in this way. That is, Proportional can be aliased to Prop in the Letters feature, but 550099BB cannot be substituted for Purple in a Color specification. For this type of thing, the \newfontfeature command should be used to declare a new, e.g., PurpleColor feature:

 $\label{lem:color} $$\operatorname{PurpleColor}_{color=550099BB}$$

Except that this example was written before support for named colours was implemented. But you get the idea.

27 Programming interface

27.1 Variables

\l_fontspec_family_tl
 \l fontspec font

In some cases, it is useful to know what the \LaTeX font family of a specific fontspec font is. After a \fontspec-like command, this is stored inside the \l_fontspec_family_tl macro. Otherwise, \LaTeX own \f@family macro can be useful here, too. The raw TeX font that is defined from the 'base' font in the family is stored in \l_fontspec_font.

\g_fontspec_encoding_tl

Package authors who need to load fonts with legacy LaTeX NFSS commands may also need to know what the default font encoding is. Since this has changed from EU1/EU2 to TU, it is best to use the variables $\g_fontspec_encoding_tl$ or \UTFencname instead.

27.2 Functions for loading new fonts and families

\fontspec_set_family:Nnn #1 : LATEX family

#2 : fontspec features

#3: font name

Defines a new NFSS family from given $\langle features \rangle$ and $\langle font \rangle$, and stores the family name in the variable $\langle family \rangle$. This font family can then be selected with standard MFEX commands $\text{fontfamily}\{\langle family \rangle\}\$ selectfont. See the standard fontspec user commands for applications of this function.

\fontspec_set_fontface:NNnn

#1 : primitive font

#2 : LaTEX family #3 : fontspec features

#4 : font name

Variant of the above in which the primitive TEX font command is stored in the variable $\langle primitive\ font \rangle$. If a family is loaded (with bold and italic shapes) the primitive font command will only select the regular face. This feature is designed for Lagrange programmers who need to perform subsequent font-related tests on the $\langle primitive\ font \rangle$.

27.3 Conditionals

The following functions in expl3 syntax may be used for writing code that interfaces with fontspec-loaded fonts. The following conditionals are all provided in TF, T, and F forms.

27.3.1 Querying font families

\fontspec_font_if_exist:nTF

Test whether the 'font name' (#1) exists or is loadable. The syntax of #1 is a restricted/simplified version of fontspec's usual font loading syntax; fonts to be loaded by filename are detected by the presence of an appropriate extension (.otf, etc.), and paths should be included inline. E.g.:

```
\fontspec_font_if_exist:nTF {cmr10}{T}{F} \fontspec_font_if_exist:nTF {Times~ New~ Roman}{T}{F} \fontspec_font_if_exist:nTF {texgyrepagella-regular.otf}{T}{F} \fontspec_font_if_exist:nTF {/Users/will/Library/Fonts/CODE2000.TTF}{T}{F}
```

The synonym \IfFontExistsTF is provided for 'document authors'.

\fontspec_if_fontspec_font:TF

Test whether the currently selected font has been loaded by fontspec.

\fontspec_if_opentype:TF

Test whether the currently selected font is an OpenType font. Always true for LuaTeX fonts.

27.3.2 Availability of features

\fontspec_if_aat_feature:nnTF

Test whether the currently selected font contains the AAT feature (#1,#2).

 $\verb|\fontspec_if_feature:nTF| \\$

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

\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:nnnTF { Returns false if the font is not loaded by fontspec or is not an OpenType font.
\fontspec_if_script:nTF	Test whether the currently selected font contains the raw OpenType script #1. E.g.: \fontspec_if_script:nTF {latn} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.
\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.
\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 {cyrl} {SRB} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.
	27.3.3 Currently selected features
ntspec_if_current_feature:nTF	Test whether the currently loaded font is using the specified raw OpenType feature tag #1. The tag string #1 should be prefixed with + to query an active feature, and with a – (hyphen) to query a disabled feature.
ontspec_if_current_script:nTF	Test whether the currently loaded font is using the specified raw OpenType script tag #1.
tspec_if_current_language:nTF	Test whether the currently loaded font is using the specified raw OpenType language

Part IX

Implementation

28 Loading

```
The expl3 module is fontspec.
 Check engine and load specific modules. For LuaTeX, load luaotfload.
 3\sys_if_engine_luatex:T
 4 { \RequirePackage{luaotfload}
      \directlua{require("fontspec")}
      \RequirePackageWithOptions{fontspec-luatex} \endinput }
 7\sys_if_engine_xetex:T
 8 { \RequirePackageWithOptions{fontspec-xetex} \endinput }
If not one of the above, error:
 9\msg_new:nnn {fontspec} {cannot-use-pdftex}
10 {
   The~ fontspec~ package~ requires~ either~ XeTeX~ or~ LuaTeX.\\\
   You~ must~ change~ your~ typesetting~ engine~ to,~ e.g.,~ "xelatex"~ or~ "lu-
  alatex" instead~ of~ plain~ "latex"~ or~ "pdflatex".
14 \msg_fatal:nn {fontspec} {cannot-use-pdftex}
15 \endinput
16 (/load)
```

29 Declaration of variables and functions

```
17 (*fontspec)
```

Booleans

firsttime 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 occuring per-shape this no longer needs to happen; this is indicated by the 'firsttime' conditional.

```
18 bool_new:N \l_@@_firsttime_bool
19 bool_new:N \l_@@_nobf_bool
20 bool_new:N \l_@@_noit_bool
21 bool_new:N \l_@@_nosc_bool
```

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

```
22\bool_new:N \l_@@_check_bool
23\cs_new:Npn \FontspecSetCheckBoolTrue { \bool_set_true:N \l_@@_check_bool }
24\cs_new:Npn \FontspecSetCheckBoolFalse { \bool_set_false:N \l_@@_check_bool }
```

```
25 \bool_new:N \l_@@_tfm_bool
26 \bool_new:N \l_@@_atsui_bool
27 \bool_new:N \l_@@_ot_bool
28 \bool_new:N \l_@@_mm_bool
29 \bool_new:N \l_@@_graphite_bool
```

For dealing with legacy maths:

```
30 \bool_new:N \g_@@_math_euler_bool
31 \bool_new:N \g_@@_math_lucida_bool
32 \bool_new:N \g_@@_pkg_euler_loaded_bool
```

For package options:

```
33 \bool_new:N \g_@@_cfg_bool
34 \bool_new:N \g_@@_math_bool
35 \bool_new:N \g_@@_euenc_bool
36 \bool_new:N \l_@@_disable_defaults_bool
37 \bool_new:N \l_@@_alias_bool
38 \bool_new:N \l_@@_external_bool
39 \bool_new:N \l_@@_never_check_bool
40 \bool_new:N \l_@@_defining_encoding_bool
41 \bool_new:N \l_@@_script_exist_bool
```

Counters

```
42 \int_new:N \l_@@_script_int
43 \int_new:N \l_@@_language_int
44 \int_new:N \l_@@_strnum_int
45 \int_new:N \l_@@_tmp_int
46 \int_new:N \l_@@_em_int
47 \int_new:N \l_@@_emdef_int
```

Floating point

Dimensions

```
50 \dim_new:N \l_@@_tmpa_dim
51 \dim_new:N \l_@@_tmpb_dim
52 \dim_new:N \l_@@_tmpc_dim
53 \seq_new:N \g_@@_bf_series_seq
```

Comma lists

```
54\clist_new:N \g_@@_default_fontopts_clist
55\clist_new:N \g_@@_all_keyval_modules_clist
56\clist_set:Nn \l_@@_sizefeat_clist {Size={-}}
```

Property lists

```
57 \prop_new:N \g_@@_fontopts_prop
58 \prop_new:N \l_@@_nfss_prop
59 \prop_new:N \l_@@_nfssfont_prop
```

```
60 \prop_new:N \g_@@_OT_features_prop
61 \prop_new:N \g_@@_all_opentype_feature_names_prop
62 \prop_new:N \g_@@_em_prop
Token lists
63 \text{ lnew:N } g_00_\text{mathrm_tl}
64 \tl_new:N \g_@@_bfmathrm_tl
65 \tl_new:N \g_00_mathsf_tl
66 \tl_new:N \g_@@_mathtt_tl
67 \text{ }\label_tl
68 \tl_new:N \l_@@_fake_slant_tl
69 \tl_new:N \l_@@_fake_embolden_tl
70 \tl_new:N \l_@@_fontname_up_tl
71 \tl_new:N \l_@@_fontname_bf_tl
72 \text{lnew:N l_00_fontname_it_tl}
73 \tl_new:N \l_@@_fontname_bfit_tl
74 \tl_new:N \l_@@_fontname_sl_tl
75 \tl new:N \l @@ fontname bfsl tl
76 \tl_new:N \l_@@_fontname_sc_tl
77 \tl_new:N \l_@@_fontfeat_up_clist
_{78}\tl_new:N \l_@@_fontfeat_bf_clist
_{79} = ... 1_00_fontfeat_it_clist
80 \tl_new:N \l_@@_fontfeat_bfit_clist
82 \times 1_new:N \l_@@_fontfeat_bfsl_clist
83 \text{l_new:N l_00_fontfeat_sc_clist}
84 \tl_new:N \l_@@_script_name_tl
85 \tl_new:N \l_fontspec_script_tl
86 \tl_new:N \l_@@_lang_name_tl
87 \tl_new:N \l_fontspec_lang_tl
88 \tl_new:N \g_@@_hexcol_tl
89 \tl_new:N \g_@@_opacity_tl
90 \tl_set:Nn \g_@@_hexcol_tl {000000}
91\tl_set:Nn \g_@@_opacity_tl {FF~}
92 \tl_set: Nn \g_@@_postadjust_tl { \l_@@_wordspace_adjust_tl \l_@@_punctspace_adjust_tl }
29.1 Generic functions
93 \cs_new:Nn \@@_keys_set_known:nnN
94 {
95 (debug) \typeout{:::: Keys~set:~{#1}~{#2} }
96 \keys set known:nnN {#1} {#2} #3
```

97 (debug) \typeout{:::: Leftover:~{#3} }

99 \cs_generate_variant:Nn \@@_keys_set_known:nnN {nx}

98 }

\@@_head_ii:n Expands to the first two \(\lambda items\rangle\) of #1.

\@@_keys_set_known:nnN

```
roo \cs_set:Npn \@@_head_ii:n #1 { \@@_head_ii:w #1 *** \q_stop}
                          ror\cs_set:Npn \@@_head_ii:w #1#2#3 \q_stop { #1#2 }
                          102 \cs_generate_variant:Nn \@@_head_ii:n {o}
\@@_int_mult_truncate:Nn Missing in expl3, IMO.
                          103 \cs_new:Nn \@@_int_mult_truncate:Nn
                                \int_set:Nn #1 { \__dim_eval:w #2 #1 \__dim_eval_end: }
                          106
                              }
                          29.2 expl3 variants
                          107 \cs_generate_variant:Nn \int_set:Nn {Nv}
                          108 \cs_generate_variant:Nn \keys_set:nn {nx}
                          109 \cs_generate_variant: Nn \keys_set_known:nnN {nx}
                          110 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
                          iii \cs generate variant:Nn \prop put:Nnn {NxV}
                          112 \cs_generate_variant:Nn \prop_gput:Nnn {NxV}
                          113 \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
                          114 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
                          115 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
                          116 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
                          rr7 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
                          rr8 \cs_generate_variant:Nn \tl_if_empty:nF {x}
                          rrg \cs_generate_variant:Nn \tl_if_empty:nF {f}
                          120 \cs generate variant:Nn \tl if eq:nnT {ox}
                          121 \cs_generate_variant:Nn \tl_replace_all:Nnn {Nnx}
                          122 (/fontspec)
                                Error/warning/info messages
                          123 (*fontspec)
                             Shorthands for messages:
                          124 \cs_new:Npn \@@_error:n
                                                         { \msg_error:nn
                                                                             {fontspec} }
                          125 \cs_new:Npn \@@_error:nn
                                                         { \msg_error:nnn
                                                                             {fontspec} }
                                                         { \msg_error:nnx
                          126 \cs_new:Npn \@@_error:nx
                                                                             {fontspec} }
                          127 \cs_new:Npn \@@_warning:n { \msg_warning:nn
                                                                             {fontspec} }
                          128\cs_new:Npn \@@_warning:nx { \msg_warning:nnx {fontspec} }
                          129 \cs_new:Npn \@@_warning:nxx { \msg_warning:nxx {fontspec} }
                          130 \cs new:Npn \@@ info:n
                                                        { \msg_info:nn
                                                                             {fontspec} }
                          131 \cs_new:Npn \@@_info:nx
                                                         { \msg_info:nnx
                                                                             {fontspec} }
                          132 \cs_new:Npn \@@_info:nxx
                                                        { \msg_info:nnxx
                                                                             {fontspec} }
                          133 \cs_new:Npn \@@_trace:n
                                                         { \msg_trace:nn
                                                                             {fontspec} }
                             Allow messages to be written with spaces acting as normal:
                          134 \cs generate variant: Nn \msg new:nnn {nnx}
                          135 \cs_generate_variant:Nn \msg_new:nnnn {nnxx}
                          136 \cs_new:Nn \@@_msg_new:nnn
                          137 { \msg_new:nnx {#1} {#2} { \tl_trim_spaces:n {#3} } }
```

139 { \msg_new:nnxx {#1} {#2} { \tl_trim_spaces:n {#3} } { \tl_trim_spaces:n {#4} } }

138 \cs_new:Nn \@@_msg_new:nnnn

```
140 \char_set_catcode_space:n {32}
```

30.1 Errors

```
r4r \@@_msg_new:nnn {fontspec} {only-inside-encdef}
142 {
143 \exp_not:N#1can only be used in the second argument
144 to \string\DeclareUnicodeEncoding.
146 \@@_msg_new:nnn {fontspec} {only-import-tu}
148 The "\string\ImportEncoding" command can only take "TU" as an argument at this stage.
149 }
150 \@@_msg_new:nnn {fontspec} {no-size-info}
151 {
152 Size information must be supplied.
For example, SizeFeatures={Size={8-12},...}.
155 \@@_msg_new:nnnn {fontspec} {font-not-found}
    The font "#1" cannot be found.
157
158 }
159 {
160 A font might not be found for many reasons.
_{161} Check the spelling, where the font is installed etc. etc.\\\
162 When in doubt, ask someone for help!
163 }
164 \@@_msg_new:nnnn {fontspec} {rename-feature-not-exist}
166 The feature #1 doesn't appear to be defined.
167 }
168 {
169 It looks like you're trying to rename a feature that doesn't exist.
171 \@@_msg_new:nnn {fontspec} {no-glyph}
172 {
     '\l_fontspec_fontname_tl' does not contain glyph #1.
173
174 }
175 \@@_msg_new:nnnn {fontspec} {euler-too-late}
176 {
    The euler package must be loaded BEFORE fontspec.
177
178 }
179 {
180 fontspec only overwrites euler's attempt to
_{\rm I8I} define the maths text fonts if fontspec is
182 loaded after euler. Type <return> to proceed
183 with incorrect \string\mathit, \string\mathbf, etc.
184 }
185 \@@_msg_new:nnnn {fontspec} {no-xcolor}
186 {
187 Cannot load named colours without the xcolor package.
188 }
```

```
189 {
190 Sorry, I can't do anything to help. Instead of loading
    the color package, use xcolor instead.
192 }
193 \@@_msg_new:nnnn {fontspec} {unknown-color-model}
194 {
    Error loading colour `#1'; unknown colour model.
195
196 }
197 {
198 Sorry, I can't do anything to help. Please report this error
    to my developer with a minimal example that causes the problem.
201 \@@_msg_new:nnnn {fontspec} {not-in-addfontfeatures}
    The "#1" font feature cannot be used in \string\addfontfeatures.
203
204 }
205 {
    This is due to how TeX loads fonts; such settings
206
    are global so adding them mid-document within a group causes
    confusion. You'll need to define multiple font families to achieve
    what you want.
209
210 }
30.2
       Warnings
211 \@@_msg_new:nnn {fontspec} {tu-clash}
213 I have found the tuenc.def encoding definition file but the TU encoding is not
    defined by the LaTeX2e kernel; attempting to correct but you really should up-
to the latest version of LaTeX2e.
216 }
217 \@@_msg_new:nnn {fontspec} {tu-missing}
219 The TU encoding seems to be missing; please update to the latest version of La-
  TeX2e.
220 }
221 \@@_msg_new:nnn {fontspec} {addfontfeatures-ignored}
222 {
   \string\addfontfeature (s) ignored \msg_line_context:;
224 it cannot be used with a font that wasn't selected by a fontspec command.\
    The current font is "\use:c{font@name}".\\
    \int_compare:nTF { \clist_count:n {#1} = 1 }
      { The requested feature is "#1". }
      { The requested features are "#1". }
229
230 }
231 \@@_msg_new:nnn {fontspec} {feature-option-overwrite}
232 {
    Option '#2' of font feature '#1' overwritten.
234 }
```

235 \@@_msg_new:nnn {fontspec} {script-not-exist-latn}

```
236 {
   Font '\l_fontspec_fontname_tl' does not contain script '#1'.\\
238
    'Latin' script used instead.
239 }
240 \@@_msg_new:nnn {fontspec} {script-not-exist}
241 {
242
    Font '\l_fontspec_fontname_tl' does not contain script '#1'.
243 }
244 \@@_msg_new:nnn {fontspec} {aat-feature-not-exist}
245 {
    '\l_keys_key_tl=\l_keys_value_tl' feature not supported
246
   for AAT font '\l_fontspec_fontname_tl'.
248 }
251 AAT feature '\l keys key tl=\l keys value tl' (#1) not available
in font '\l fontspec fontname tl'.
253 }
254 \@@_msg_new:nnn {fontspec} {icu-feature-not-exist}
255 {
    '\l_keys_key_tl=\l_keys_value_tl' feature not supported
256
    for OpenType font '\l_fontspec_fontname_tl'
257
258 }
259 \@@_msg_new:nnn {fontspec} {icu-feature-not-exist-in-font}
260 {
    OpenType feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
261
    for font '\l_fontspec_fontname_tl'
    with script '\l_@@_script_name_tl' and language '\l_@@_lang_name_tl'.
263
264 }
265 \@@_msg_new:nnn {fontspec} {no-opticals}
266 {
    '\l_fontspec_fontname_tl' doesn't appear to have an Optical Size axis.
267
268 }
269 \@@_msg_new:nnn {fontspec} {language-not-exist}
271 Language '#1' not available
272 for font '\l fontspec fontname tl'
273 with script '\l_@@_script_name_tl'.\\
    'Default' language used instead.
274
275 }
276 \@@_msg_new:nnn {fontspec} {only-xetex-feature}
277 {
278 Ignored XeTeX only feature: '#1'.
279 }
280 \@@_msg_new:nnn {fontspec} {only-luatex-feature}
281 {
    Ignored LuaTeX only feature: '#1'.
282
283 }
284 \@@_msg_new:nnn {fontspec} {no-mapping}
285 {
286 Input mapping not (yet?) supported in LuaTeX.
```

```
287 }
288 \@@_msg_new:nnn {fontspec} {no-mapping-ligtex}
289 {
    Input mapping not (yet?) supported in LuaTeX.\\
291
    Use "Ligatures=TeX" instead of "Mapping=tex-text".
292 }
293 \@@_msg_new:nnn {fontspec} {cm-default-obsolete}
294 {
    The "cm-default" package option is obsolete.
295
296 }
297 \@@_msg_new:nnn {fontspec} {fakebold-only-xetex}
298 {
_{299} The "FakeBold" and "AutoFakeBold" options are only available with XeLaTeX.\\
300 Option ignored.
301 }
302 \00 msg new:nnn {fontspec} {font-index-needs-ttc}
303 {
304 The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.
305 Feature ignored.
306 }
309 The "#1" feature cannot be deactivated. Request ignored.
310 }
30.3 Info messages
311 \@@_msg_new:nnn {fontspec} {defining-font}
312 {
313 Font family '\l_fontspec_family_tl' created for font '#2'
    with options [\l_@@_all_features_clist].\\
314
315
    This font family consists of the following NFSS series/shapes:\\
    \l_fontspec_defined_shapes_tl
317
318 }
319 \@@_msg_new:nnn {fontspec} {no-font-shape}
320 {
321 Could not resolve font "#1" (it probably doesn't exist).
322 }
323 \@@_msg_new:nnn {fontspec} {set-scale}
324 {
^{325} \l_fontspec_fontname_tl\space scale = \l_@0_scale_tl.
327 \@@_msg_new:nnn {fontspec} {setup-math}
329 Adjusting the maths setup (use [no-math] to avoid this).
330 }
331 \@@_msg_new:nnn {fontspec} {no-scripts}
333 Font "\l_fontspec_fontname_tl" does not contain any OpenType `Script' in-
  formation.
334 }
```

```
_{335}\ensuremath{\mbox{00_msg_new:nnn}} {fontspec} {opa-twice}
336 {
337
    Opacity set twice, in both Colour and Opacity. \\
338 Using specification "Opacity=#1".
339 }
340 \@@_msg_new:nnn {fontspec} {opa-twice-col}
341 {
342 Opacity set twice, in both Opacity and Colour.
    Using an opacity specification in hex of "#1/FF".
344 }
_{345}\ensuremath{\mbox{00_msg\_new:nnn}} {fontspec} {bad-colour}
346 {
347 Bad colour declaration "#1".
348 Colour must be one of:\\
349 * a named xcolor colour\\
350 * a six-digit hex colour RRGGBB\\
* an eight-digit hex colour RRGGBBTT with opacity
352 }
    Reset 'space' behaviour:
353 \char_set_catcode_ignore:n {32}
354 (/fontspec)
```

31 Opening code

31.1 Package options

```
355 \DeclareOption{cm-default}
356 { \@@_warning:n {cm-default-obsolete} }
357 \DeclareOption{math}{\bool_set_true:N \g_@@_math_bool}
358 \DeclareOption{no-math}{\bool_set_false:N \g_@@_math_bool}
359 \DeclareOption{config}{\bool_set_true:N \g_@@_cfg_bool}
360 \DeclareOption{no-config}{\bool_set_false:N \g_@@_cfg_bool}
361 \DeclareOption{euenc}{\bool_set_true:N \g_@@_euenc_bool}
{\tt 362 \backslash DeclareOption\{tuenc\}\{\backslash bool\_set\_false: N \backslash g\_@@\_euenc\_bool\}}
363 \DeclareOption{quiet}
364 {
365 \msg_redirect_module:nnn { fontspec } { warning } { info }
366 \msg_redirect_module:nnn { fontspec } { info } { none }
368 \DeclareOption{silent}
369 {
     \msg redirect module:nnn { fontspec } { warning } { none }
    \msg_redirect_module:nnn { fontspec } { info } { none }
372 }
373 \ExecuteOptions{config,math,tuenc}
374 \ProcessOptions*
```

31.2 Encodings

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

```
375 \bool_if:NF \g_@@_euenc_bool
```

```
376
       \file_if_exist:nTF {tuenc.def}
377
378
           \cs_if_exist:cF {T@TU}
379
380
381
              \@@_warning:n {tu-clash}
382
              \DeclareFontEncoding{TU}{}{}
              383
384
        }
385
386
           \@@_warning:n {tu-missing}
387
           \bool_set_true:N \g_@@_euenc_bool
388
389
390
391 \bool_if:NTF \g_@@_euenc_bool
393 (xetexx)
             \tl_set:Nn \g_fontspec_encoding_tl {EU1}
394 (luatex)
             \tl_set:Nn \g_fontspec_encoding_tl {EU2}
395 }
    { \tl_set:Nn \g_fontspec_encoding_tl { TU } }
397 \tl_set:Nn \rmdefault {lmr}
398 \tl_set:Nn \sfdefault {lmss}
399 \tl_set:Nn \ttdefault {lmtt}
400 \RequirePackage[\g_fontspec_encoding_tl]{fontenc}
401 \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:

```
402 \tl_if_in:NnT \@filelist {.cls} { \normalsize }
```

Dealing with a couple of the problems introduced by babel:

```
403 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
404 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
405 \AtBeginDocument
406 {
407 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
408 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
409 }
```

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.

```
410 \bool_if:NT \g_@@_euenc_bool
411 {
412 \langle luatex \ \cs_set_eq:NN \fontspec_tmp: \XeTeXpicfile
413 \langle luatex \ \cs_set:Npn \XeTeXpicfile {}
414 \RequirePackage{xunicode}
415 \langle luatex \ \cs_set_eq:NN \XeTeXpicfile \fontspec_tmp:
416 }
```

32 expl3 interface for font loading

```
n,\@@_primitive_font_gset:Nnn
                                                                                    417 \cs_set:Npn \00_primitive_font_set:Nnn #1#2#3
                                                                                    418 {
                                                                                                       \font #1 = #2 ~at~ #3 \scan_stop:
                                                                                   419
                                                                                   420
                                                                                   421 \cs_set:Npn \@@_primitive_font_gset:Nnn #1#2#3
                                                                                                      \global \font #1 = #2 ~at~ #3 \scan_stop:
                                                                                   423
                                                                                   424
ont_suppress_not_found_error:
                                                                                   425 \cs_set:Npn \@@_font_suppress_not_found_error:
                                                                                                       \int_set_eq:NN \xetex_suppressfontnotfounderror:D \c_one
                                                                                   427
                                                                                   428
                                                                              [ pTF]@_primitive_font_if_null:N
                                                                                   429 \prg_set_conditional: Nnn \00_primitive_font_if_null: N {p,TF,T,F}
                                                                                   430 {
                                                                                                      \ifx #1 \nullfont
                                                                                   431
                                                                                                            \prg_return_true:
                                                                                    432
                                                                                                      \else
                                                                                                            \prg_return_false:
                                                                                   434
                                                                                                       \fi
                                                                                   435
                                                                                              }
                                                                                   436
                                                                             [ TF]@_primitive_font_if_exist:n
                                                                                   437 \prg_set_conditional:Nnn \@@_primitive_font_if_exist:n {TF,T,F}
                                                                                   438
                                                                                                       \group_begin:
                                                                                   439
                                                                                                             \00 font suppress not found error:
                                                                                   440
                                                                                                             \@@_primitive_font_set:Nnn \l_@@_primitive_font {#1} {10pt}
                                                                                   441
                                                                                                            \@@_primitive_font_if_null:NTF \l_@@_primitive_font
                                                                                   442
                                                                                                                 { \group_end: \prg_return_false: }
                                                                                   443
                                                                                                                  { \group_end: \prg_return_true: }
                                                                                    444
                                                                                                }
                                                                                    445
tive_font_glyph_if_exist:NnTF
                                                                                   {\tt 446 \prg\_new\_conditional:Nnn \end{conditional:Nnn \end{conditional:
                                                                                   447
                                                                                                       \etex_iffontchar:D #1 #2 \scan_stop:
                                                                                   448
                                                                                                            \prg_return_true:
                                                                                   449
                                                                                                       \else:
                                                                                   450
                                                                                                            \prg_return_false:
                                                                                   451
                                                                                                       \fi:
                                                                                   452
                                                                                   453
```

33 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 35.1 on page 93.

33.0.1 Font selection

\fontspec

This is the main command of the package that selects fonts with various features. It takes two arguments: the font name and the optional requested features of that font. Then this new font family is selected.

```
454 \NewDocumentCommand \fontspec { O{} m O{} }
455 {
456 \fontspec_set_family:Nnn \f0family {#1,#3} {#2}
457 \fontencoding { \l_00_nfss_enc_tl }
458 \selectfont
459 \ignorespaces
460 }
```

\setmainfont

The following three macros perform equivalent operations setting the default font for a particular family: 'roman', sans serif, or typewriter (monospaced). I end them with \normalfont so that if they're used in the document, the change registers immediately.

```
461 \DeclareDocumentCommand \setmainfont { O{} m O{} }
           462 {
               \tl_set_eq:NN \rmdefault \g_@@_rmfamily_family
               \use:x { \exp_not:n { \DeclareRobustCommand \rmfamily }
           465
           466
                 \exp not:N \fontencoding { \l @@ nfss enc tl }
           467
                 \exp not:N \fontfamily { \g @@ rmfamily family }
           468
                 \exp_not:N \selectfont
           469
           470
               }
           471
               \str_if_eq_x:nnT {\familydefault} {\rmdefault}
           472
                 { \tl_set_eq:NN \encodingdefault \l_@0_nfss_enc_tl }
           473
               \normalfont
           474
               \ignorespaces
           475
           476 }
\setsansfont
           _{477}\\ ( O{} m O{} )
           478 {
               479
               \t_{g_00_sffamily_family}
           480
               \use:x { \exp_not:n { \DeclareRobustCommand \sffamily }
           481
           482
                 \exp not:N \fontencoding { \l @@ nfss enc tl }
           483
                 \exp not:N \fontfamily { \g @@ sffamily family }
           484
                 \exp not:N \selectfont
           485
           186
```

```
487
                     \str_if_eq_x:nnT {\familydefault} {\sfdefault}
                488
                489
                       { \tl_set_eq:NN \encodingdefault \l_@@_nfss_enc_tl }
                490
                     \normalfont
                491
                     \ignorespaces
                492 }
  \setmonofont
                493 \DeclareDocumentCommand \setmonofont { O{} m O{} }
                494 {
                     \fontspec set family: Nnn \g @@ ttfamily family {#1,#3} {#2}
                495
                     \tl_set_eq:NN \ttdefault \g_@@_ttfamily_family
                496
                     \use:x { \exp_not:n { \DeclareRobustCommand \ttfamily }
                497
                498
                       \exp_not:N \fontencoding { \l_@@_nfss_enc_tl }
                499
                       \exp_not:N \fontfamily { \g_@@_ttfamily_family }
                500
                       \verb|\exp_not:N \edges | Selectfont|
                501
                502
                503
                504
                     \str_if_eq_x:nnT {\familydefault} {\ttdefault}
                       { \tl_set_eq:NN \encodingdefault \l_@0_nfss_enc_tl }
                505
                     \normalfont
                506
                     \ignorespaces
                507
                508 }
                This is the old name for \setmainfont, retained ad infinitum for backwards compati-
 \setromanfont
                 bility. It was deprecated in 2010.
                509 \cs_set_eq:NN \setromanfont \setmainfont
                These commands are analogous to \setmainfont and others, but for selecting the
    \setmathrm
                 font used for \mathrm, etc. They can only be used in the preamble of the document.
    \setmathsf
\setboldmathrm
                 \setboldmathrm is used for specifying which fonts should be used in \boldmath.
    \setmathtt
                510 \DeclareDocumentCommand \setmathrm { O{} m O{} }
                511 {
                    \fontspec_set_family:Nnn \g_@@_mathrm_tl {#1} {#2}
                512
                513 }
                514
```

515 \DeclareDocumentCommand \setboldmathrm { O{} m O{} } 516 { \fontspec_set_family: Nnn \g_@@_bfmathrm_tl {#1} {#2}

518 } 519

 $_{520}$ \DeclareDocumentCommand \setmathsf { O{} m O{} }

\fontspec_set_family:Nnn \g_@@_mathsf_tl {#1} {#2} 523 } 524 525 \DeclareDocumentCommand \setmathtt { O{} m O{} }

\fontspec_set_family:\nn \g_@@_mathtt_tl \{\#1\} \{\#2\} 527 528 }

```
529 \@onlypreamble\setmathrm
530 \@onlypreamble\setboldmathrm
531 \@onlypreamble\setmathsf
532 \@onlypreamble\setmathtt
```

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

\newfontfamily

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

```
536 \DeclareDocumentCommand \newfontfamily { m O{} m O{} }
537 {
    538
    \use:x
539
540
     \exp_not:N \DeclareRobustCommand \exp_not:N #1
541
542
       \exp not:N \fontfamily { \use:c \{g @@ \cs to str:N #1 family\} \}
543
       \exp not:N \fontencoding { \l @@ nfss enc tl }
544
       \exp not:N \selectfont
545
546
     }
547
548 }
```

\newfontface

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

```
_{549}\ \DeclareDocumentCommand \newfontface { m O{} m O{} } $$ 550 { $$ 551 $$ \left[ BoldFont={},ItalicFont={},SmallCapsFont={},#2,#4 ] {#3} $$ 552 }
```

33.0.2 Font feature selection

\defaultfontfeatures

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

```
553 \DeclareDocumentCommand \defaultfontfeatures { t+ o m }
554 {
555 \IfNoValueTF {#2}
556 { \@@_set_default_features:nn {#1} {#3} }
557 { \@@_set_font_default_features:nnn {#1} {#2} {#3} }
558 \ignorespaces
```

```
559 }
560 \cs_new:\n \@@_set_default_features:nn
561 {
562 \IfBooleanTF {#1} \clist_put_right:\n \clist_set:\n
563 \g_@@_default_fontopts_clist {#2}
564 }
```

The optional argument specifies a font identifier. 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.

```
565 \cs_new: Nn \@@_set_font_default_features:nnn
566 {
    \clist map inline:nn {#2}
567
568
      \tl if single:nTF {##1}
569
       570
        { \ensuremath{\mbox{00\_sanitise\_fontname:Nn \l_00\_tmp\_tl {##1}}}
571
572
       \IfBooleanTF {#1}
573
574
         \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
575
          { \tl_clear:N \l_@@_tmpb_tl }
576
        \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
577
         \prop_gput:NVV \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
578
        }
579
        {
580
        \tl_if_empty:nTF {#3}
581
          { \prop_gremove:NV \g_@@_fontopts_prop \l_@@_tmp_tl }
582
                            \g_00_{\text{fontopts\_prop}} \l_00_{\text{tmp\_tl}} \{\#_3,} \}
583
          { \prop_put:NVn
584
585
```

\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.

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

```
592
        \group_begin:
593
          \keys_set_known:nnN {fontspec-addfeatures} {#1} \l_@@_tmp_tl
          \prop_get:cnN {g_@@_ \f@family _prop} {options} \l_@@_options_tl
594
          \prop_get:cnN {g_@@_ \f@family _prop} {fontname} \l_@@_fontname_tl
595
          \bool_set_true:N \l_@@_disable_defaults_bool
596
597 (debug)
                      \label{typeout} $$ \end{area} $$ \sup_{0 \le s \in L_{n}} \frac{1_{00\_options\_tl}, \#_1}{1_{00\_options\_tl}}, \#_1$ } $$
          \use:x
598
           {
599
            \@@_select_font_family:nn
600
              { \l_@@_options_tl , #1 } {\l_@@_fontname_tl}
601
602
603
        \group_end:
       \fontfamily\l_fontspec_family_tl\selectfont
604
605
606
        \@@_warning:nx {addfontfeatures-ignored} {#1}
607
608
     \ignorespaces
609
610 }
611 \cs_set_eq:NN \addfontfeature \addfontfeatures
```

33.0.3 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.

\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.

```
622 \DeclareDocumentCommand \newAATfeature \{mmmm\}
623 \{
624 \keys_if_exist:nnF \{ fontspec \} \{\frac{\mathref{\pi}}{1\}\}
625 \{ \QQ_define_aat_feature_group:n \{\frac{\mathref{\pi}}{1\}} \{\frac{\mathref{\pi}}{2\}}
626 \keys_if_choice_exist:nnnT \{\frac{\mathref{\pi}}{1\}} \{\frac{\mathref{\pi}}{2\}}
627 \{ \QQ_warning:nxx \{\mathref{\pi}}{2\}} \{\mathref{\pi}}{2\}}
628 \QQ_define_aat_feature:nnnn \{\frac{\mathref{\pi}}{1\}} \{\frac{\mathref{\pi}}{2\}}
629 \}
```

\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.

```
630 \DeclareDocumentCommand \newopentypefeature {mmm}
                          631 {
                               \keys_if_exist:nnF { fontspec / options } {#1}
                          632
                                 { \@@_define_opentype_feature_group:n {#1} }
                               \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
                                 { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
                               \ensuremath{\texttt{00\_define\_opentype\_feature:nnn}} \fi \fi \fi \fi \fi
                          637 }
         \newICUfeature
                          638 \cs set eq:NN \newICUfeature \newopentypefeature % deprecated
      \aliasfontfeature User commands for renaming font features and font feature options.
                          639 \DeclareDocumentCommand \aliasfontfeature {mm}
                               \bool_set_false:N \l_@@_alias_bool
                          641
                          642
                               \clist_map_inline:nn
                          643
                                { fontspec, fontspec-opentype, fontspec-aat, fontspec-preparse, fontspec-
                             preparse-external,
                                  fontspec-preparse-nested, fontspec-renderer }
                          645
                          646
                          647
                          648
                                  \keys_if_exist:nnT {##1} {#1}
                          649
                                      \bool set true: N \l @@ alias bool
                          650
                                      \keys define:nn {##1}
                          651
                                        { \#2 .code:n = { \keys set:nn {\#\#1} { \#1 = {\#\#\#1} } } }
                          652
                          653
                                }
                          654
                          655
                               \bool_if:NF \l_@@_alias_bool
                                 { \@@_warning:nx {rename-feature-not-exist} {#1} }
                          658 }
\aliasfontfeatureoption
                          659 \DeclareDocumentCommand \aliasfontfeatureoption {mmm}
                          66o {
                          661
                               \bool set false: N \l @@ alias bool
                          662
                               \clist_map_inline: Nn \g_@@_all_keyval_modules_clist
                          663
                          664
                                {
                          665
                                  \keys_if_exist:nnT {##1} {#1}
                          666
                          667
                                      \bool_set_true:N \l_@@_alias_bool
                          668
                                      \keys_define:nn { \##1 / \#1 } { \#3 .meta:n = {\#2} }
                          669
                          670
                                }
                          671
                          672
                               \bool_if:NF \l_@@_alias_bool
```

```
{ \@@_warning:nx {rename-feature-not-exist} {#1} }
                       674
                       675 }
                       Mostly used internally, but also possibly useful for users, to define new OpenType
        \newfontscript
                        'scripts', mapping logical names to OpenType script tags.
                       676 \DeclareDocumentCommand \newfontscript {mm}
                       678 \fontspec_new_script:nn {#1} {#2}
                       679 }
                       Mostly used internally, but also possibly useful for users, to define new OpenType
      \newfontlanguage
                       'languages', mapping logical names to OpenType language tags.
                       680 \DeclareDocumentCommand \newfontlanguage {mm}
                       682
                            \fontspec_new_lang:nn {#1} {#2}
                       683 }
                       dfont would never be uppercase, right?
\DeclareFontsExtensions
                       684 \DeclareDocumentCommand \DeclareFontsExtensions {m}
                            \clist set:Nn \l @@ extensions clist { #1 }
                           \tl_remove_all:Nn \l_@@_extensions_clist {~}
                       687
                       688 }
                       689 \DeclareFontsExtensions{.otf,.ttf,.OTF,.TTF,.ttc,.TTC,.dfont}
     \IfFontFeaturesTF
                       690 \DeclareDocumentCommand \IfFontFeatureActiveTF {mmm}
                       691 {
                       692 (debug)
                                    \label{typeout} $$ \sup_{n=1}^{\#1}{\#2}{\#3}} $$
                       693 (debug)
                              \00_if_font_feature:nTF {#1} {#2} {#3}
                       696 \prg_new_conditional:Nnn \@@_if_font_feature:n {TF}
                       697
                       698
                              \tl_gclear:N \g_@@_single_feat_tl
                              \group_begin:
                       699
                                \@@_font_suppress_not_found_error:
                       700
                                \@@ init:
                       701
                                \bool set true:N \l @@ ot bool
                       702
                                \bool set true:N \1 @@ never check bool
                       703
                                \bool_set_false:N \l_@@_firsttime_bool
                       704
                                \clist_clear:N \l_@@_fontfeat_clist
                       705
                                \@0_get_features:Nn \l_@0_rawfeatures_sclist {#1}
                       706
                              \group_end:
                       707
                       708
                                    \typeout{:::> \exp_not:N\l_@@_rawfeatures_sclist->~{\l_@@_rawfeatures_sclist}}
                       709 (debug)
                                    710 (debug)
                       711
                              \tl if empty:NTF \g @@ single feat tl { \prg return false: }
                       712
                       713
                                  \exp_args:NV \fontspec_if_current_feature:nTF \g_@@_single_feat_tl
                       714
```

```
715 { \prg_return_true: } { \prg_return_false: }
716 }
717 }
```

34 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.

```
718 \prg_new_conditional:Nnn \fontspec_if_fontspec_font: {TF,T,F}
719 {
720 \cs_if_exist:cTF {g_@@_ \f@family _prop} \prg_return_true: \prg_return_false:
721 }
```

\fontspec if aat feature:nnTF

Conditional to test if the currently selected font contains the AAT feature (#1,#2).

```
722 \prg_new_conditional:Nnn \fontspec_if_aat_feature:nn {TF,T,F}
723 {
     \fontspec_if_fontspec_font:TF
724
725
       \prop_get:cnN {g_@@_ \f@family _prop} {fontdef} \l_@@_fontdef_tl
726
       \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
727
       \bool_if:NTF \l_@@_atsui_bool
728
        {
729
         \@@_make_AAT_feature_string:nnTF {#1}{#2}
730
           \prg return true: \prg return false:
731
732
        {
733
         \prg_return_false:
734
735
      }
736
      {
737
738
       \prg_return_false:
      }
739
740
```

\fontspec_if_opentype:TF

Test whether the currently selected font is an OpenType font. Always true for LuaTeX fonts.

```
741 \prg_new_conditional:Nnn \fontspec_if_opentype: {TF,T,F}
742 {
743 \fontspec_if_fontspec_font:TF
744 {
745 \prop_get:cnN {g_@@_ \f@family _prop} {fontdef} \l_@@_fontdef_tl
```

```
746  \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
747  \@@_set_font_type:
748  \bool_if:NTF \l_@@_ot_bool \prg_return_true: \prg_return_false:
749  }
750  {
751  \prg_return_false:
752  }
753 }
```

\fontspec_if_feature:nTF

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

```
754 \prg_new_conditional: Nnn \fontspec_if_feature:n {TF,T,F}
755 {
                 \fontspec_if_fontspec_font:TF
756
                     {
757
                         \label{lem:converse_general} $$ \operatorname{g_00}_{f0family\_prop} {fontdef} \label{lem:converse} $$ \end{figure} $$ \operatorname{fontdef}_{t1} $$
758
                         \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
759
760
                         \@@_set_font_type:
                         \bool_if:NTF \l_@@_ot_bool
761
762
                                 \prop_get:cnN {g_@@_ \f@family _prop} {script-num} \l_@@_tmp_tl
763
                                \label{local_script_int} $$ \left( \frac{1_00_script_int {\l_00_tmp_tl}}{} \right) $$
764
765
                                \label{lem:cnn} $$ \operatorname{g_00}_{\operatorname{prop}} {\operatorname{lang-num}} \label{lang-num} $$ \end{tikzpicture} $$ \operatorname{lang-num} \end{tikzpicture} $$ \end{tikzpicture} $$ \operatorname{lang-num} \end{tikzpicture} $$ \end{tikzpicture}
766
                                \int_set:Nn \l_@@_language_int {\l_@@_tmp_tl}
767
768
                                \prop_get:cnN {g_00_ \f0family _prop} {script-tag} \l_fontspec_script_tl
769
                                \prop_get:cnN {g_00_ \f0family _prop} {lang-tag} \l_fontspec_lang_tl
771
                                \@@_check_ot_feat:nTF {#1} {\prg_return_true:} {\prg_return_false:}
772
773
                             {
774
                                  \prg_return_false:
775
776
                     }
777
                     {
778
779
                         \prg_return_false:
                     }
78o
781 }
```

 $\verb|\fontspec_if_feature:nnnTF| \\$

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 {la Returns false if the font is not loaded by fontspec or is not an OpenType font.

```
782 \prg_new_conditional:Nnn \fontspec_if_feature:nnn {TF,T,F}
783 {
784 \fontspec_if_fontspec_font:TF
785 {
786 \prop_get:cnN {g_@@_ \f@family _prop} {fontdef} \l_@@_fontdef_tl
787 \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
788 \@@_set_font_type:
```

```
\bool_if:NTF \l_@@_ot_bool
789
790
791
       \@@_iv_str_to_num:Nn \l_@@_language_int {#2}
792
       \@@_check_ot_feat:nTF {#3} \prg_return_true: \prg_return_false:
793
794
       { \prg_return_false: }
795
796
     { \prg_return_false: }
797
798 }
```

\fontspec_if_script:nTF

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

```
799 \prg_new_conditional: Nnn \fontspec_if_script:n {TF,T,F}
800 €
8от
      \fontspec_if_fontspec_font:TF
802
        \label{lem:converse_general} $$ \operatorname{g_00_ fofamily prop} {fontdef} \label{lem:converse_general} $$ \operatorname{g_00_ fontdef_tl} $$
803
804
        \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
        \@@_set_font_type:
805
        \bool_if:NTF \l_@@_ot_bool
806
807
           \@@_check_script:nTF {#1} \prg_return_true: \prg_return_false:
808
800
         { \prg_return_false: }
810
8тт
       { \prg_return_false: }
812
813 }
```

\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.

```
814 \prg_new_conditional: Nnn \fontspec_if_language:n {TF,T,F}
815 {
                           \fontspec_if_fontspec_font:TF
816
817
                                        \prop_get:cnN {g_@@_ \f@family _prop} {fontdef} \l_@@_fontdef_tl
818
                                       \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
819
                                        \@@_set_font_type:
820
821
                                        \bool if:NTF \l @@ ot bool
                                            {
822
                                                   \label{lem:cnn} $$ \operatorname{g_00} \left( \operatorname{prop} \left( \operatorname{script-num} \right) \right) = 0. $$ \operatorname{cnn} \left( \operatorname{script-num} \right) = 0. $$
823
                                                  \label{local_script_int} $$ \left( \frac{1_00_tmp_tl}{2} \right) $$ int_set: Nn \left( \frac{1_00_tmp_tl}{2} \right) $$ 
824
                                                  \prop_get:cnN {g_@@_ \f@family _prop} {script-tag} \l_fontspec_script_tl
825
826
                                                  \label{lang:nTF $$\#_1$ \operatorname{prg\_return\_true: \operatorname{prg\_return\_false:}} $$
827
828
                                             }
829
                                             { \prg_return_false: }
830
831
                                 { \prg_return_false: }
```

```
832 }
```

\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 {cyrl} {SRB} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.

```
833 \prg new conditional: Nnn \fontspec if language:nn {TF,T,F}
834 {
     \fontspec_if_fontspec_font:TF
835
836
        \label{lem:converse_general} $$ \operatorname{g_00_ fofamily prop} {fontdef} \label{lem:converse_general} $$ \operatorname{g_00_ fontdef_tl} $$
837
        \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
838
        \@@_set_font_type:
839
840
        \bool_if:NTF \l_@@_ot_bool
841
          \tl_set:Nn \l_fontspec_script_tl {#1}
842
          \@@_iv_str_to_num:Nn \l_@@_script_int {#1}
843
          \@@_check_lang:nTF {#2} \prg_return_true: \prg_return_false:
844
845
846
         { \prg_return_false: }
847
848
       { \prg_return_false: }
849 }
```

ontspec_if_current_script:nTF

Test whether the currently loaded font is using the specified raw OpenType script tag

```
850 \prg_new_conditional:Nnn \fontspec_if_current_script:n {TF,T,F}
851 {
     \fontspec_if_fontspec_font:TF
852
853
        \label{lem:converse_general} $$ \operatorname{g_00} \left( \frac{g_0 - g_0}{g_0} \right) = \frac{1}{2} \left( \frac{g_0 - g_0}{g_0} \right) $$
854
        \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
855
        \@@_set_font_type:
856
        \bool_if:NTF \l_@@_ot_bool
857
858
          \prop_get:cnN {g_@@_ \f@family _prop} {script-tag} \l_@@_tmp_tl
859
          \str_if_eq:nVTF {#1} \l_@@_tmp_tl
860
             {\prg_return_true:} {\prg_return_false:}
861
862
         { \prg_return_false: }
863
       }
864
       { \prg_return_false: }
865
866 }
```

tspec_if_current_language:nTF

Test whether the currently loaded font is using the specified raw OpenType language tag #1.

```
867 \prg_new_conditional:Nnn \fontspec_if_current_language:n {TF,T,F}
868 {
869 \fontspec_if_fontspec_font:TF
870 {
871 \prop_get:cnN {g_@@_ \f@family _prop} {fontdef} \l_@@_fontdef_tl
872 \@@_primitive_font_set:Nnn \l_fontspec_font {\l_@@_fontdef_tl} {\f@size pt}
```

```
\@@_set_font_type:
                                                                    873
                                                                                     \bool_if:NTF \l_@@_ot_bool
                                                                     874
                                                                     875
                                                                                           \label{lem:cnn} $$ \operatorname{g_00} \left( \frac{g_0}{g_0} \right) - \left( \frac{g_0}{g_0} \right) = 0. $$
                                                                     876
                                                                     877
                                                                                          \str_if_eq:nVTF {#1} \l_@@_tmp_tl
                                                                     878
                                                                                               {\prg_return_true:} {\prg_return_false:}
                                                                     879
                                                                     88o
                                                                                        { \prg_return_false: }
                                                                     881
                                                                     882
                                                                                   { \prg_return_false: }
                                                                     883 }
                                                                    #1 : family
       \fontspec_set_family:Nnn
                                                                      #2 : fontspec features
                                                                      #3: font name
                                                                              Defines a new font family from given \langle features \rangle and \langle font \rangle, and stores the name
                                                                      in the variable \langle family \rangle. See the standard fontspec user commands for applications of
                                                                              We want to store the actual name of the font family within the \langle family \rangle variable
                                                                      because the actual LTFX family name is automatically generated by fontspec and it's
                                                                      easier to keep it that way.
                                                                              Please use \fontspec set family: Nnn instead of \@@ select font family:nn,
                                                                      which may change in the future.
                                                                     884 \cs_new:Nn \fontspec_set_family:Nnn
                                                                     885 {
                                                                               \tl_set:Nn \l_@0_family_label_tl { #1 }
                                                                    887
                                                                                \@@_select_font_family:nn {#2}{#3}
                                                                                \tl_set_eq:NN #1 \l_fontspec_family_tl
                                                                    890 \cs generate variant: Nn \fontspec set family: Nnn {c}
\fontspec_set_fontface:NNnn
                                                                    891 \cs_new:Nn \fontspec_set_fontface:NNnn
                                                                    892 {
                                                                               \tl_set:Nn \l_@@_family_label_tl { #1 }
                                                                             \@@_select_font_family:nn {#3}{#4}
                                                                            \tl_set_eq:NN #1 \l_fontspec_font
                                                                     896 \tl_set_eq:NN #2 \l_fontspec_family_tl
                                                                     897 }
     \fontspec_font_if_exist:n
                                                                     898 \texttt{\prg\_new\_conditional:Nnn \fontspec\_font\_if\_exist:n \{TF,T,F\}}
                                                                     899
                                                                                      \group_begin:
                                                                     900
                                                                                           \@0_init:
                                                                     901
                                                                                          \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
                                                                     902
                                                                                          \label{lem:construct_font_call:nn $$\#_1$ {} $$ \end{tikzpicture} $$$ \end{tikzpicture} $$ \end{tikzpicture} $$$ \end{tikz
                                                                     903
                                                                                               { \group end: \prg return true: }
                                                                     904
                                                                                               { \group_end: \prg_return_false: }
                                                                     905
                                                                     906
```

907 \cs set eq:NN \IfFontExistsTF \fontspec font if exist:nTF

ntspec_if_current_feature:nTF

Test whether the currently loaded font is using the specified raw OpenType feature tag #1

35 Internals

35.1 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 \footspec -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 \footspec_family_tl . The \footspec_foots (globally) stored in \footspec_foots .

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.
- \1_@@_basename_tl is the (immutable) original argument used for *-replacing.
- \l_fontspec_font is the plain TEX font of the upright font requested.

```
914 \cs_new_protected:Nn \@@_select_font_family:nn
915 {
916 (debug)\typeout{^^J^^J:::::::::::::::::::^]:: fontspec_select:nn~ {#1}~ {#2} }
     \group begin:
     \@@_font_suppress_not_found_error:
918
    \@@ init:
919
920
     \@@ sanitise_fontname:Nn \l_fontspec_fontname_tl
                                                          {#2}
921
     \@@_sanitise_fontname: Nn \l_@@_fontname_up_tl {#2}
922
     \@@_sanitise_fontname:Nn \l_@@_basename_tl
                                                          {#2}
923
     \@@_if_detect_external:nT {#2}
925
926
     { \keys_set:nn {fontspec-preparse-external} {Path} }
927
    \@@_init_ttc:n {#2}
    \@@_load_external_fontoptions:\n\l_fontspec_fontname_tl \\ \#2\}
929
930
```

```
\@@_extract_all_features:n {#1}
     \tl_set:Nx \l_@@_fontid_tl { \tl_to_str:N \l_fontspec_fontname_tl-:-\tl_to_str:N \l_@@_all_:
933
934 (debug)\typeout{fontid: \l_@@_fontid_tl}
935
     \@@_preparse_features:
936
     \@@_load_font:
937
     \@@_set_scriptlang:
938
     \@@_get_features:Nn \l_@@_rawfeatures_sclist {}
939
     \bool_set_false:N \l_@@_firsttime_bool
940
941
     \@@_save_family_needed:nTF {#2}
942
943
        \00_save_family:nn {#1} {#2}
944
945 (debug) \@@_warning:nxx {defining-font} {#1} {#2}
946
947
_{948}\,\langle \text{debug}\rangle \typeout{Font~ family~ already~ defined.}
      }
949
     \group_end:
950
951 }
```

\fontspec_select:nn

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

```
952 \cs_set_eq:NN \fontspec_select:nn \@@_select_font_family:nn
```

\@@_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.

```
953 \cs_new:Nn \@@_sanitise_fontname:Nn
954 {
    \tl_set:Nx #1 {#2}
956 (luatex) \tl_remove_all:Nn #1 {~}
     \clist_map_inline:Nn \l_@@_extensions_clist
957
958
        \tl_if_in:NnT #1 {##1}
959
          {
960
            \tl_remove_once:Nn #1 {##1}
961
            \tl set:Nn \l @@ extension tl {##1}
962
             \clist_map_break:
963
964
      }
965
966 }
```

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

```
967 \prg_new_conditional:Nnn \@@_if_detect_external:n {T}
969 \langle debug \rangle \typeout{:: @@_if_detect_external:n { \exp_not:n {#1} } }
   \clist_map_inline:Nn \l_@@_extensions_clist
971
```

```
\exp_args:Nx % <- this should be handled earlier
                                                                                                                     973
                                                                                                                     974
                                                                                                                                                \tl_if_in:nnT {#1 <= end_of_string} {##1 <= end_of_string}
                                                                                                                                                        { \bool_set_true:N \l_@@_tmpa_bool \clist_map_break: }
                                                                                                                     975
                                                                                                                     976
                                                                                                                                         \bool_if:NTF \l_@@_tmpa_bool \prg_return_true: \prg_return_false:
                                                                                                                     977
                                                                                                                     978 }
                                                         \@@_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.
                                                                                                                      979 \cs_new:Nn \@@_init ttc:n
                                                                                                                     980 {
                                                                                                                                        \str_if_eq_x:nnT { \str_lower_case:f {\l_@0_extension_tl} } {.ttc}
                                                                                                                     981
                                                                                                                     982
                                                                                                                                                 \@@_sanitise_fontname:Nn \l_@@_fontname_it_tl
                                                                                                                                                                                                                                                                                                                                              {#1}
                                                                                                                     983
                                                                                                                                                \@@_sanitise_fontname:Nn \l_@@_fontname_bf_tl
                                                                                                                                                                                                                                                                                                                                              {#1}
                                                                                                                     984
                                                                                                                                                \@@_sanitise_fontname:Nn \l_@@_fontname_bfit_tl {#1}
                                                                                                                      985
                                                                                                                     986
                                                                                                                     987 }
                                                                                                                     Load a possible .fontspec font configuration file. This file could set font-specific op-
_load_external_fontoptions:Nn
                                                                                                                        tions for the font about to be loaded.
                                                                                                                      988 \cs_new:Nn \@@_load_external_fontoptions:Nn
                                                                                                                     989 {
                                                                                                                     990 (debug) \typeout{:: @@_load_external_fontoptions:Nn \exp_not:N #1 {#2} }
                                                                                                                                        \@@_sanitise_fontname:Nn #1 {#2}
                                                                                                                                         \tl_set:Nx \l_@@_ext_filename_tl {#1.fontspec}
                                                                                                                      992
                                                                                                                                         \tl remove all:Nn \l @@ ext filename tl {~}
                                                                                                                     993
                                                                                                                                         \prop_if_in:NVF \g_@@_fontopts_prop #1
                                                                                                                     994
                                                                                                                     995
                                                                                                                                                \exp_args:No \file_if_exist:nT { \l_@0_ext_filename_tl }
                                                                                                                      996
                                                                                                                                                    { \left\{ \right. \left. \right. \right. \right. \left. \left. \right. \left. \left. \right. \left. \left. \right. \left. \left. \right. \left. \left. \right. \left. \left. \right. \left. \left. \right. \left. \left. \right. \left. \left. \right. \left. \right.
                                                                                                                      997
                                                                                                                                           }
                                                                                                                      998
                                                                                                                     999 }
             \@@_extract_all_features:
                                                                                                                   1000 \cs_new:Nn \@@_extract_all_features:n
                                                                                                                   1001 {
                                                                                                                   1002 (debug) \typeout{:: @@_extract_all_features:n { \unexpanded {#1} } }
                                                                                                                                         \bool_if:NTF \l_@@_disable_defaults_bool
                                                                                                                   1004
                                                                                                                                                \clist_set:Nx \l_@0_all_features_clist {#1}
                                                                                                                   1005
                                                                                                                  1006
                                                                                                                   1007
                                                                                                                                                 \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
                                                                                                                   1008
                                                                                                                                                   { \clist_clear:N \l_@@_fontopts_clist }
                                                                                                                   1009
                                                                                                                   1010
                                                                                                                                                \prop_get:NVNF \g_@@_fontopts_prop \l_@@_family_label_tl \l_@@_family_fontopts_clist
                                                                                                                   1011
                                                                                                                                                    { \clist_clear:N \l_@@_family_fontopts_clist }
                                                                                                                   1012
                                                                                                                   1013
                                                                                                                                                \tl_clear:N \l_@@_family_label_tl
                                                                                                                  1014
```

\bool_set_false:N \l_@@_tmpa_bool

972

\@@_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.

```
1024 \cs_new:Nn \@@_preparse_features:
1025 {
1026 \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.

```
1027
1028 \@@_keys_set_known:nxN {fontspec-preparse-external}
1029 { \l_@@_all_features_clist }
1030 \l_@@_keys_leftover_clist
```

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.

```
1032 \tl_set_eq:NN \l_fontspec_fontname_tl \l_@@_fontname_up_tl
1033 \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_keys_leftover_clist}
1034 \l_@@_keys_leftover_clist
1035 \@@_keys_set_known:nxN {fontspec-preparse} {\l_@@_keys_leftover_clist}
1036 \l_@@_fontfeat_clist
1037 }
```

 $\00_{\odd}$ font:

1048

{ $\ensuremath{\verb|@0_construct_font_call:nn|} { \ensuremath{\verb|l_00_fontname_up_tl|} } {} } } } }$

```
\l_fontspec_font % this is necessary for LuaLaTeX to check the scripts prop-
                                 erly
                              1050 }
  #2 : Extension
                               #3: TTC Index
                               #4: Renderer
                               #5 : Optical size
                               #6 : Font features
                                  We check if (Font features) are empty and if so don't add in the separator colon.
                              1051\cs_set:Nn \@@_construct_font_call:nnnnnn
                              1052 {
                                   " \@@_fontname_wrap:n { #1 #2 #3 } #4 #5
                              1053
                                     \str_if_eq_x:nnF {#6}{} {:#6} "
                              1054
                              1055
                               In practice, we don't use the six-argument version, since most arguments are con-
                               structed on-the-fly:
                              1056 \cs_set:Nn \@@_construct_font_call:nn
                             1057 {
                                   \@@ construct font call:nnnnnn
                             1058
                                     {#1}
                              1059
                                     \1 @@ extension tl
                              T060
                                     \l_@@_ttc_index_tl
                              то6т
                                     \l_fontspec_renderer_tl
                              1062
                                     \l_@@_optical_size_tl
                              1063
                              1064
                                     {#2}
                              1065 }
                              The \@@_fontname_wrap:n command takes the font name and either passes it
nt_is_file:,\@@_font_is_name:
                               through unchanged or wraps it in the syntax for loading a font 'by filename'. XaTeX's
                               syntax is followed since luaotfload provides compatibility.
                              1066 \cs_new: Nn \@@_font_is_name:
                              1067
                              1068
                                     \cs_set_eq:NN \00_fontname_wrap:n \use:n
                             1069
                             1070 \cs new:Nn \@@ font is file:
                             1071 {
                             1072 (xetexx)
                                            \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
                             1073 (luatex)
                                            \cs_set:Npn \@@_fontname_wrap:n ##1 { file:\l_@@_font_path_tl ##1 }
                             1074
                                  }
                              Only necessary for OpenType fonts. First check if the font supports scripts, then apply
         \00 set scriptlang:
                               defaults if none are explicitly requested. Similarly with the language settings.
                              1075 \cs_new:Nn \@@_set_scriptlang:
                              1076 {
                                   \bool if:NT \l @@ firsttime bool
                             1077
                             1078
                                     \tl_if_empty:NTF \l_@@_script_name_tl
                             1079
```

то8о

{

```
\@@_check_script:nTF {latn}
1081
1082
            \tl_set:Nn \l_@@_script_name_tl {Latin}
1083
1084
            \tl_if_empty:NT \l_@@_lang_name_tl
1085
1086
              \tl_set:Nn \l_@@_lang_name_tl {Default}
1087
            \keys_set:nx {fontspec-opentype} {Script=\l_@0_script_name_tl}
1088
            \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
1089
1090
1091
            \@@_info:n {no-scripts}
1092
           }
1093
         }
1094
1095
          \tl_if_empty:NT \l_@@_lang_name_tl
1096
1097
            \tl_set:Nn \l_@@_lang_name_tl {Default}
1098
1099
          \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
TIOO
          \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
IIOI
1102
1103
       }
1104 }
```

\@@_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.

```
1105 \cs_set:Nn \@@_get_features:Nn
1106 {
rio7 (debug) \typeout{:: @0_get_features:Nn \exp_not:N #1 { \exp_not:n {#2} } }
                         \@@_init_fontface:
1108
                         \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#2}
1109
                                   \l_@@_keys_leftover_clist
1110
1111
                         \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
1112 (*xetexx)
                         \bool_if:NTF \l_@@_ot_bool
1113
1114
1115 (debug) \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
                                            \keys_set:nx {fontspec-opentype} {\l_@@_keys_leftover_clist}
1116
1117
                                   {
1118
1119 (debug)
                                                    \typeout{::: Setting~ keys~ for~ AAT~ font~ features:~"\l_@@_keys_leftover_clist"}
                                            \bool_if:NT \l_@@_atsui_bool
1120
                                                     { \ensuremath{\mbox{\mbox{$\setminus$} \ensuremath{\mbox{$\setminus$} \ensuremath{
1121
                                   }
1123 (/xetexx)
1124 (*luatex)
```

\@@_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.

```
1135 \prg_new_conditional:Nnn \@@_save_family_needed:n {TF}
1136 {
1137
1138 (debug) \typeout{save~ family:~ #1}
rr39 (debug) \typeout{== fontid_tl: "\l_@@_fontid_tl".}
1140
     \cs_if_exist:NT \l_@@_nfss_fam_tl
1141
1142
       \cs_set_eq:cN {g_@@_UID_\l_@@_fontid_tl} \l_@@_nfss_fam_tl
1143
1144
     \cs_if_exist:cF \{g_00_UID_\l_00_fontid_tl\}
1145
1146
       \% The font name is fully expanded, in case it's defined in terms of macros, be-
1147
   fore having its spaces zapped:
       \tl_set:Nx \l_@@_tmp_tl {#1}
       \tl_remove_all:Nn \l_@@_tmp_tl {~}
1149
1150
       1151
        1152
        { \int new:c
                       {g_000\_family\_ \l_000\_tmp_tl \_int} }
1153
1154
       \t_gset:cx {g_@@_UID_\l_@@_fontid_tl}
1155
1156
         \l_00_{\rm tmp\_tl} \ ( int_use:c \{g_00_{\rm family} \ l_00_{\rm tmp\_tl} \ int \} )
1157
        }
1158
1159
     \tl_gset:Nv \l_fontspec_family_tl {g_@@_UID_\l_@@_fontid_tl}
1160
     \cs_if_exist:cTF {g_@@_ \l_fontspec_family_tl _prop}
1161
1162
       \prg_return_false: \prg_return_true:
1163 }
```

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

```
1164 \cs_new:Nn \@@_save_family:nn
1165 {
```

```
\@@_save_fontinfo:n {#2}
                    1166
                    1167
                            \@@_find_autofonts:
                    1168
                            \DeclareFontFamily{\l_@@_nfss_enc_tl}{\l_fontspec_family_tl}{}
                    1169
                            \@@_set_faces:
                    1170
                            \@@_info:nxx {defining-font} {#1} {#2}
                    1171
\@@_save_fontinfo:n Saves the relevant font information for future processing.
                    1172 \cs_new: Nn \@@_save_fontinfo:n
                          \prop_new:c {g_@@_ \l_fontspec_family_tl _prop}
                    1174
                          \prop_gput:cnx {g_00_ \l_fontspec_family_tl _prop} {fontname} { #1 }
                    1175
                          \prop_gput:cnx {g_@@_ \l_fontspec_family_tl _prop} {options} { \l_@@_all_features_clist }
                    1176
                          \prop_gput:cnx {g_@@_ \l_fontspec_family_tl _prop} {fontdef}
                    1177
                    1178
                            \@@_construct_font_call:nn {\l_fontspec_fontname_tl}
                    1179
                              { \l_@@_pre_feat_sclist \l_@@_rawfeatures_sclist }
                    1180
                    1181
                          \prop_gput:cnV {g_@@_ \l_fontspec_family_tl _prop} {script-num} \l_@@_script_int
                    1182
                          \prop_gput:cnV {g_@@_ \l_fontspec_family_tl _prop} {lang-num} \l_@@_language_int
                    1183
                          \prop_gput:cnV {g_@@_ \l_fontspec_family_tl _prop} {script-tag} \l_fontspec_script_tl
                          \prop_gput:cnV {g_@@_ \l_fontspec_family_tl _prop} {lang-tag} \l_fontspec_lang_tl
                    1185
                    1186 }
```

35.2 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.

 $\verb|\@0_find_autofonts:|$

```
1187 \cs_new: Nn \@@_find_autofonts:
                              \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
1189
1190
                                         \@@ set autofont:Nnn \l @@ fontname bfit tl {\l @@ fontname it tl} {/B}
1191
                                        \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
1192
                                        \label{lem:lem:local_set_autofont:Nnn l_00_fontname_bfit_tl {\l_fontspec_fontname_tl} {\label{lem:lem:local_bfit_bfit} } % \end{substitute} % $$ $$ \end{substitute} $$ $$ \end{substitute} $$ \end{substitute} $$ \end{substitute} $$ \end{substitute} $$ $$ \end{substitute} $$ \end{subst
1193
1194
1195
                              \bool_if:NF \l_@@_nobf_bool
1196
1197
                                        \label{lem:local_cont} $$ 0_{\text{set}_autofont:Nnn} \local_fontname_bf_tl {\local_fontspec_fontname_tl} {\local_fontspec_fontname_tl} $$
1198
1199
1200
                              \bool_if:NF \l_@@_noit_bool
1201
1202
```

```
\@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspec_fontname_tl} {/I}
                                                              1203
                                                                           }
                                                              1204
                                                                         \label{local_set_autofont:Nnn l_00_fontname_bfsl_tl {l_00_fontname_sl_tl} {/B}} $$ $$ (B) $
                                                             1207 }
                              \@@_set_faces:
                                                             1208\cs_new:Nn \@@_set_faces:
                                                             1209 {
                                                                         \@@_add_nfssfont:nnnn \mddefault \updefault \l_fontspec_fontname_tl
                                                                                                                                                                                                                                    \l_@@_fontfeat_up_@
                                                             1210
                                                                         \@@_add_nfssfont:nnnn \bfdefault \updefault \l_@@_fontname_bf_tl
                                                                                                                                                                                                                       \l_@@_fontfeat_bf_clist
                                                                         \@@_add_nfssfont:nnnn \mddefault \itdefault \l_@@_fontname_it_tl
                                                                                                                                                                                                                       \l_@@_fontfeat_it_clist
                                                                         \@@_add_nfssfont:nnnn \mddefault \sldefault \l_@@_fontname_sl_tl
                                                                                                                                                                                                                       \l_@@_fontfeat_sl_clist
                                                                         \@@ add nfssfont:nnnn \bfdefault \itdefault \l @@ fontname bfit tl \l @@ fontfeat bfit clis
                                                             1214
                                                                         \@@ add nfssfont:nnnn \bfdefault \sldefault \l @@ fontname bfsl tl \l @@ fontfeat bfsl clis
                                                             1215
                                                             1216
                                                                         \prop_map_inline: Nn \l_00_nfssfont_prop { \00_set_faces_aux:nnnnn ##2 }
                                                             1217
                                                             1218 }
                                                             1219 \cs_new:Nn \@@_set_faces_aux:nnnnn
                                                             1220 {
                                                                        \fontspec_complete_fontname: Nn \l_@@_curr_fontname_tl {#3}
                                                                       \@@_make_font_shapes:\nnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
                                                              1223 }
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").
                                                             r224 \cs_set:Nn \fontspec_complete_fontname:Nn
                                                             1225 {
                                                                        \tl_set:Nx #1 {#2}
                                                             1226
                                                                        \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
                                                             1228 (luatex) \tl_remove_all:Nn #1 {~}
                                                              1229 }
               \@@_add_nfssfont:nnnn #1 : series
                                                                #2 : shape
                                                                #3: fontname
                                                                #4 : fontspec features
                                                             1230 \cs_new: Nn \@@_add_nfssfont:nnnn
                                                             1231 {
                                                                         tl_set:Nx \l_@@_this_font_tl {#_3}
                                                             1232
                                                             1233
                                                                         \tl_if_empty:xTF {#4}
                                                              1234
                                                                           { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
                                                              1235
                                                                           { \00_{keys\_set\_known:nxN} \{fontspec\_preparse\_nested\} \{\#4\} \1_00_{tmp\_tl} \}
                                                              1236
                                                              1237
                                                                         \tl_if_empty:NF \l_@@_this_font_tl
                                                              1238
                                                              1239
                                                                             \prop put:Nxx \l @@ nfssfont prop {#1/#2}
                                                             1240
                                                                                { \#1}{\#2}{\ 00 \text{ this font tl}}{\#4}{\ 00 \text{ sizefeat clist} }
                                                             1241
                                                             1242
                                                             1243 }
```

35.2.1 Fonts

\@@_set_font_type:

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.

```
1244 \cs_new: Nn \@@_set_font_type:
1245 {
1246 (debug) \typeout{:: @@_set_font_type:}
1247 (*xetexx)
      \bool_set_false:N \l_@@_tfm_bool
1248
      \bool_set_false:N \l_@@_atsui_bool
1249
      \bool_set_false:N \l_@@_ot_bool
1250
      \bool_set_false:N \l_@@_mm_bool
1251
      \bool_set_false:N \l_@@_graphite_bool
1252
      \ifcase\XeTeXfonttype\l_fontspec_font
1253
        \bool_set_true:N \l_@@_tfm_bool
1254
1255
     \or
        \bool_set_true:N \l_@@_atsui_bool
1256
        \ifnum\XeTeXcountvariations\l_fontspec_font > \c_zero
1257
          \bool_set_true:N \l_@@_mm_bool
1258
        \fi
1259
1260
        \bool_set_true:N \l_@@_ot_bool
1261
1262
```

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.

```
\tl_if_empty:NT \l_fontspec_renderer_tl
1263
1264
        \bool_if:NTF \l_@@_atsui_bool
1265
         { \tl_set:Nn \l_fontspec_renderer_tl {/AAT} }
1266
1267
1268
           \bool_if:NT \l_@@_ot_bool
1269
            { \tl_set:Nn \l_fontspec_renderer_tl {/OT} }
1270
       }
1271
1272 (/xetexx)
1273 (*luatex)
     \bool_set_true:N \l_@@_ot_bool
1275 (/luatex)
1276 }
```

 $\00_set_autofont:Nnn #1 : Font name tl$

#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.

1277 \cs_new: Nn \@@_set_autofont: Nnn

We can't match external fonts in this way (in X \exists TeX anyway; todo: test with Lua-TeX). 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.

```
1278 {
                                    \bool_if:NF \l_@@_external_bool
                              1279
                              1280
                                    \tl_if_empty:xF {#2}
                              1281
                              1282
                                      \tl_if_empty:NT #1
                              1283
                              1284
                                         \ensuremath{\texttt{00\_if\_autofont:nnTF}} \ensuremath{\texttt{\#2}} \ensuremath{\texttt{\#3}}
                              1285
                              1286
                                          { \tl set:Nx #1 {#2#3} }
                              1287
                                          { \@@ info:nx {no-font-shape} {#2#3} }
                              1288
                              T289
                              1290
                              1291
                              1292
                              1293 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
                              1294
                                    \@@_primitive_font_set:Nnn \l_tmpa_font { \@@_construct_font_call:nn {#1} {} } {\f@size pt}
                              1295
                                    \@@_primitive_font_set:Nnn \l_tmpb_font { \@@_construct_font_call:nn {#1#2} {} } {\f@size primitive_font_set:Nnn \l_tmpb_font { \@@_construct_font_call:nn {#1#2} {} } }
                              1296
                                    \str_if_eq_x:nnTF { \fontname \l_tmpa_font } { \fontname \l_tmpb_font }
                              1297
                                     { \prg_return_false: }
                              1298
                                     { \prg_return_true: }
                              1299
                              1300 }
\@@_make_font_shapes:Nnnnn
                               #1 : Font name
                               #2 : Font series
                               #3 : Font shape
                               #4 : Font features
                               #<sub>5</sub> : Size features
                                   This macro eventually uses \DeclareFontShape to define the font shape in ques-
                              1301 \cs new: Nn \@@ make font shapes: Nnnnn
                              1302 {
                                    \group_begin:
                              1303
                                       \@@_keys_set_known:nxN {fontspec-preparse-external} { #4 } \l_@@_leftover_clist
                              1304
                                      \@@_load_fontname:n {#1}
                              1305
                                      1306
                                    \group_end:
                              1307
                              1308 }
                              1309
                              1310 \cs_new: Nn \@@_load_fontname:n
                              1311 {
                                              \typeout{:: @@_load_fontname:n {#1} }
                              1312 (debug)
```

```
\@@_load_external_fontoptions:\n\l_fontspec_fontname_tl {#1}
                                                   1313
                                                                  \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
                                                   1314
                                                                    { \clist_clear:N \l_@@_fontopts_clist }
                                                   1315
                                                   1316
                                                                  \@@_primitive_font_set:Nnn \l_fontspec_font { \@@_construct_font_call:nn {\l_fontspec_font_set:Nnn \l_fontspec_font_set:Nnn \l_font_set:Nnn \l_fo
                                                                  \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-
                                                   1317
                                                          found} {#1} }
                                                  1318 }
   \verb|\@@_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 us-
                                                     ing \1_@@_nfss_tl defined above. \1_@@_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>}}.
                                                   1319 \cs_new: Nn \@@_declare_shape:nnnn
                                                  1321 (debug)\typeout{=~ declare shape:~{\l fontspec fontname tl}~{#1}~{#2}}
                                                           \tl clear:N \l @@ nfss tl
                                                             \tl_clear:N \l_@@_nfss_sc_tl
                                                  1323
                                                             \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspec_fontname_tl
                                                  1324
                                                  1325
                                                              \ensuremath{\texttt{exp\_args:Nx \clist\_map\_inline:nn \ \{\#_4\} \ \{ \ensuremath{\texttt{@0\_setup\_single\_size:nn \ \{\#_3\} \ \{\#\#_1\} \ }} }
                                                  1326
                                                  1327
                                                             \@@_declare_shapes_normal:nn {#1} {#2}
                                                  1328
                                                              \@@_declare_shapes_smcaps:nn {#1} {#2}
                                                   1329
                                                              \@@_declare_shape_slanted:nn {#1} {#2}
                                                  1330
                                                             \@@ declare shape loginfo:nn {#1} {#2}
                                                  1331
                                                  1333 \cs generate variant: Nn \00 declare shape:nnnn {nnxx}
\@@_setup_single_size:nn
                                                   1334 \cs_new:Nn \@@_setup_single_size:nn
                                                   1335
                                                                  \tl_clear:N \l_@@_size_tl
                                                   1336
                                                                  \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
                                                   1337
                                                  1338
                                                                  \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
                                                  1339
                                                                      \l_@@_sizing_leftover_clist
                                                   1340
                                                                  \tl_if_empty:NT \l_@0_size_tl { \00_error:n {no-size-info} }
                                                   1341
                                                   _{1342} \langle debug \rangle =  size: ~\l_@@_size_t1
                                                   1343
                                                                  % "normal"
                                                  1344
                                                                  \@@_load_fontname:n {\l_@@_sizedfont_tl}
                                                  1345
                                                                  \@@_setup_nfss:Nnnn \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
                                                                               \typeout{===~ sized~ font:~ \l_@@_sizedfont_tl}
                                                  1347 (debug)
                                                  1348
                                                                  % small caps
                                                  1349
                                                                  \clist set eq:NN \l @@ fontfeat curr clist \l @@ fontfeat sc clist
                                                   1350
```

```
\verb|\bool_if:NF \l_@@\_nosc_bool|
                              1352
                              1353
                                         \tl_if_empty:NTF \l_@@_fontname_sc_tl
                              1354
                              1355
                                           \@@_make_smallcaps:TF
                              1356
                              1357
                              1358 (debug)\typeout{====~Small~ caps~ found.}
                                             \clist_put_left:\n\\l_@0_fontfeat_curr_clist {Letters=SmallCaps}
                              1359
                              1360
                              1361
                              1362 (debug)\typeout{====~Small~ caps~ not~ found.}
                                             \bool_set_true:N \l_@@_nosc_bool
                              1363
                              1364
                              1365
                                          { \00_load_fontname:n {\l_00_fontname_sc_tl} }\% local for each size}
                              1366
                              1367
                              1368
                                       \bool_if:NF \l_@@_nosc_bool
                              1369
                              1370
                                         \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
                               1371
                                           {#1} {\l_@0_sizing_leftover_clist} {\l_@0_fontfeat_curr_clist}
                              1372
                              1373
                                    }
                              1374
         \@@_setup_nfss:Nnnn
                              1375 \cs_new: Nn \@@_setup_nfss: Nnnn
                              1377 (debug)\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}
                                     \00 get features: Nn \1 00 rawfeatures sclist { #2 , #3 , #4 }
                              1379
                              1380 (debug)\typeout{====~Gathered~features:~\l_@@_rawfeatures_sclist}
                              1381
                                    \tl_put_right:Nx #1
                              1382
                              1383
                                      <\l_@0_size_tl> \l_@0_scale_tl
                               1384
                                         \@@_construct_font_call:nn { \l_fontspec_fontname_tl }
                               1385
                                           { \l_@0_pre_feat_sclist \l_@0_rawfeatures_sclist }
                               1386
                              1387
                              1388 }
\@@_declare_shapes_normal:nn
                              1389 \cs new: Nn \@@ declare shapes normal:nn
                              1390
                                      \@@ DeclareFontShape:xxxxxx {\1 @@ nfss enc tl} {\1 fontspec family tl}
                              1391
                                         {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
                              1392
                              1393
\@@_declare_shapes_smcaps:nn
                               1394 \cs_new:Nn \@@_declare_shapes_smcaps:nn
                              1395 {
```

1351

```
\tl_if_empty:NF \l_@@_nfss_sc_tl
1396
1397
1398
         \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl} {\l_fontspec_family_tl} {#1}
           1399
1400
1401
1402
1403 \cs_new:Nn \@@_combo_sc_shape:n
1404
       \tl if exist:cTF { \@@ shape merge:nn {#1} {\scdefault} }
1405
            { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
1406
            { \scdefault }
1407
1408
     }
1409 \cs_new: Nn \@@_DeclareFontShape:nnnnnn
1410 {
1411 (debug)\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
     \group_begin:
1412
       \normalsize
1413
       \cs_undefine:c {#1/#2/#3/#4/\f@size}
1414
     \group_end:
1415
     \ensuremath{\texttt{DeclareFontShape}} \#_1 \ \#_2 \ \#_3 \ \#_4 \ \#_5 \ \#_6 \
1416
1417 }
1418\cs_generate_variant:Nn \00_DeclareFontShape:nnnnnn {xxxxxx}
```

\@@_declare_shape_slanted:nn

\@@ DeclareFontShape:nnnnnn

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.

```
1419 \cs_new: Nn \@@_declare_shape_slanted:nn
1420 {
     \bool_if:nT
1421
       {
1422
         \str if eq x p:nn {#2} {\itdefault} &&
1423
        !(\str_if_eq_x_p:nn {\itdefault} {\sldefault})
1424
       }
1425
       {
1426
        \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl}{\l_fontspec_family_tl}{#1}{\sldefault}
1427
          {<->ssub*\l_fontspec_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
1428
       }
1429
1430 }
```

\@@_declare_shape_loginfo:nn Lastly some informative messaging.

```
1431 \cs_new:\n \@@_declare_shape_loginfo:nn
1432 {
1433 \tl_gput_right:\nx \l_fontspec_defined_shapes_tl
1434 {
1435 \exp_not:n { \\ }
```

```
-~ \exp_not:N \str_case:nn {#1/#2}
1436
        {
1437
1438
           {\mddefault/\updefault} {'normal'~}
1439
           {\bfdefault/\updefault} {'bold'~}
           {\mddefault/\itdefault} {'italic'~}
1440
           {\mddefault/\sldefault} {'slanted'~}
1441
           {\bfdefault/\itdefault} {'bold~ italic'~}
1442
           {\bfdefault/\sldefault} {'bold~ slanted'~}
1443
        } (#1/#2)~
1444
       with~ NFSS~ spec.:~
1445
       \label{local_nfss_tl} $$ 1_00_nfss_tl $$
1446
       \exp_{not:n} \{ \ \ \ \}
1447
        -~ \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
1448
1449
           {\mddefault/\scdefault} {'small~ caps'~}
1450
           {\bfdefault/\scdefault} {'bold~ small~ caps'~}
1451
           {\mddefault/\itscdefault} {'italic~ small~ caps'~}
1452
           {\bfdefault/\itscdefault} {'bold~ italic~ small~ caps'~}
1453
           {\mddefault/\slscdefault} {'slanted~ small~ caps'~}
1454
           {\bfdefault/\slscdefault} {'bold~ slanted~ small~ caps'~}
1455
        1456
        with~ NFSS~ spec.:~
1457
        \l_@@_nfss_sc_tl
1458
        \tl_if_empty:fF {\l_@@_postadjust_tl}
1459
1460
          \exp_not:N \\ and~ font~ adjustment~ code: \exp_not:N \\ \l_@@_postadjust_tl
1461
1462
1463
1464 }
```

Maybe \str_if_eq_x:nnF would be better?

35.2.2 Features

\l_@@_pre_feat_sclist These are the features always applied to a font selection before other features.

```
1465 \clist_set:Nn \l_@@_pre_feat_sclist
1466 (*xetexx)
1467 {
      \bool_if:NT \l_@@_ot_bool
1468
1469
        \tl if empty:NF \l fontspec script tl
1470
1471
                   = \l_fontspec_script_tl ;
          script
1472
          language = \l_fontspec_lang_tl ;
1473
1474
1475
       }
1476 }
_{1477} \langle /xetexx \rangle
1478 (*luatex)
1479 {
1480
     mode
                = \l_fontspec_mode_tl
1481 \tl_if_empty:NF \l_fontspec_script_tl
```

```
script = \l_fontspec_script_tl ;
                                1483
                                1484
                                       language = \l_fontspec_lang_tl
                                1485
                               1486 }
                               1487 (/luatex)
    \@@_make_ot_smallcaps:TF This macro checks if the font contains small caps.
                               1488 (luatex)\cs_set:Nn \@@_make_smallcaps:TF
                               1489 (xetexx)\cs_set:Nn \@@_make_ot_smallcaps:TF
                               1490 {
                                     \00 check ot feat:nTF {smcp} {#1} {#2}
                               1491
                               1492 }
                               1493 (*xetexx)
                               1494\cs_set:Nn \@@_make_smallcaps:TF
                               1495 {
                                     \bool_if:NTF \l_@@_ot_bool
                               1496
                                       { \@@_make_ot_smallcaps:TF {#1} {#2} }
                               1497
                               1498
                                         \bool_if:NT \l_@@_atsui_bool
                               1499
                               1500
                                          { \@@_make_AAT_feature_string:nnTF {3}{3} {#1} {#2} }
                               1501
                               1502 }
                               1503 (/xetexx)
         \@@_update_featstr:n \l_@@_rawfeatures_sclist is the string used to define the list of specific font fea-
                                 tures. Each time another font feature is requested, this macro is used to add that feature
                                 to the list. Font features are separated by semicolons.
                                1504 \cs new:Nn \@@ update featstr:n
                                     {
                               1505
                               1506 (debug)
                                                       \typeout{:::: @@_update_featstr:n {#1}}
                                        \verb|\bool_if:NF \l_@@_firsttime_bool|
                               1507
                               1508
                                            \t!_gset:Nx \g_00\_single_feat_tl { #1 }
                                1509
                                                       \typeout{::::~ Adding~ feature.}
                                1510 (debug)
                                            \tl_gput_right:Nx \l_@@_rawfeatures_sclist {#1;}
                               1512
                               1513
\@@_remove_clashing_featstr:n
                                1514 \cs_new: Nn \@@_remove_clashing_featstr:n
                               1515 {
                               1516 (debug)
                                               \typeout{:::: @@_remove_clashing_featstr:n {#1}}
                                        \clist_map_inline:nn {#1}
                               1517
                               1518
                               1519 (debug)
                                                   \typeout{::::~ Removing~ feature~ "##1;"}
                                            \tl_gremove_all:Nn \l_@@_rawfeatures_sclist {##1;}
                               1520
                               1521
```

1482

1522 }

35.3 Initialisation

\@@_init: 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.)

```
1523 \cs_set:Npn \@@_init:
                   1524 {
                   _{1525}\langle debug\rangle \typeout{:: @@_init:}
                         \bool_set_false:N \l_@@_ot_bool
                         \bool_set_true:N \l_@@_firsttime_bool
                   1527
                        \@@ font is name:
                   1528
                        \tl_clear:N \l_@@_font_path_tl
                        \tl_clear:N \l_@@_optical_size_tl
                   1530
                        \tl_clear:N \l_@@_ttc_index_tl
                   1531
                         \tl_clear:N \l_fontspec_renderer_tl
                    1532
                         \tl clear: N \l fontspec defined shapes tl
                    1533
                         \tl clear:N \g @@ curr series tl
                    1534
                         \tl_gset_eq:NN \l_@0_nfss_enc_tl \g_fontspec_encoding_tl
                    1535
                    1536
                   1537 (*luatex)
                         \tl_set:Nn \l_fontspec_mode_tl {node}
                   1538
                         \int_set:Nn \luatex_prehyphenchar:D { `\- } % fixme
                    1539
                         \int_zero:N \luatex_posthyphenchar:D
                                                                        % fixme
                         \int_zero:N \luatex_preexhyphenchar:D
                                                                        % fixme
                         \int_zero:N \luatex_postexhyphenchar:D
                                                                        % fixme
                    1543 (/luatex)
\@@_init_fontface: Executed in \@@_get_features:Nn.
                   1545 \cs_new: Nn \@@_init_fontface:
                   1546
                            \tl clear:N \l @@ rawfeatures sclist
                   1547
                            \tl clear:N \l @@ scale tl
                    1548
                            \tl_set_eq:NN \l_@@_opacity_tl \g_@@_opacity_tl
                    1549
                            \tl_set_eq:NN \l_@@_hexcol_tl \g_@@_hexcol_tl
                    1550
                            \tl_set_eq:NN \l_@@_postadjust_tl \g_@@_postadjust_tl
                    1551
                           \tl_clear:N \l_@@_wordspace_adjust_tl
                    1552
                    1553
                            \tl_clear:N \l_@@_punctspace_adjust_tl
                    1554
```

35.4 Miscellaneous

 $\verb|\@@_iv_str_to_num:Nn|$

This macro takes a four character string and converts it to the numerical representation required for X₃T_EX OpenType script/language/feature purposes. The output is stored in #1.

The reason it's ugly is because the input can be of the form of any of these: 'abcd', 'abc', 'abc', 'abc', 'abc', 'etc. (It is assumed the first two chars are always not spaces.) So this macro reads in the string, delimited by a space; this input is padded with \@empty s and anything beyond four chars is snipped. The \@empty s then are used to reconstruct the spaces in the string to number calculation.

```
1555 \cs_set:Nn \@@_iv_str_to_num:Nn
```

```
1556 {
1557
      \@@_iv_str_to_num:w #1 \q_nil #2 \c_empty_tl \c_empty_tl \q_nil
1558 }
1559 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
1560 {
1561
     \int_set:Nn #1
1562
1563
          `#2 * "1000000
       + `#3 * "10000
1564
       + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
1565
       + \ifx \c_empty_tl \#_5 32 \else `\#_5 \fi
1566
1567
1568 }
1569\cs_generate_variant:Nn \@@_iv_str_to_num:Nn {No}
```

36 OpenType definitions code

```
fine_opentype_feature_group:n
                                  1570 \cs_new:Nn \@@_define_opentype_feature_group:n
                                           \keys define:nn {fontspec-opentype} { #1 .multichoice: }
                                  1573
define_opentype_feature:nnnnn #1 : Feature key
                                   #2 : Feature option val
                                   #3 : Check feature — leave empty for no check
                                   #4 : Exact tag string to activate — leave empty for disable only
                                   #<sub>5</sub> : Tags to remove (clist)
                                  1574 \cs_new:Nn \@@_feat_prop_add:nn
                                       {
                                  1575
                                           \tl_if_empty:nF {#1}
                                  1576
                                  1577
                                             \prop_if_in:NnF \g_@@_OT_features_prop {#1}
                                  1578
                                                  \prop_gput:Nnn \g_00_OT_features_prop {#1} {#2}
                                  1581
                                  1582
                                  1583
                                  1584 \cs_new:Nn \@@_define_opentype_feature:nnnnn
                                  1585 {
                                        \label{eq:condition} $$ \ensuremath{\mbox{\tt 00\_feat\_prop\_add:nn $\{\#_3\} $\{\#_1\,=\,\#_2\}$} $$
                                  1586
                                        \keys_define:nn {fontspec-opentype}
                                  1587
                                  1588
                                          \#_1/\#_2 .code:n = { \@@_make_OT_feature:nnn {\#_3} {\#_4} {\#_5} }
                                  1589
                                  1590
                                  1591 }
ine_opentype_onoffreset:nnnnn #1 : Feature key
                                   #2 : Feature option val
                                   #3 : Check feature
```

```
1592 \cs_new: Nn \@@_feat_off:n {#10ff}
                                1593 \cs_new:Nn \@@_feat_reset:n {#1Reset}
                                1594\cs_new:Nn \@@_define_opentype_onoffreset:nnnnn
                               1595 {
                               1596
                                     \exp_args:\nx \@@_define_opentype_feature:nnnnn \\ \#1\} \\ \#3\} \\ \#4\} \\ \#5\}
                                     \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_off:n {#2} } {#3} {-
                               1597
                                   #4} {}
                                     \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {} {+#4,-
                                1598
                                   #4}
                                1599 }
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)
                                1600 \cs new: Nn \00 define opentype onreset:nnnnn
                                1601 {
                                     \exp_args:\nx \00_define_opentype_feature:nnnnn \{\pmu_1\} \{\pmu_2\} \{\pmu_3\} \{\pmu_4\} \{\pmu_5\}
                                     \exp_args:Nnx \00_define_opentype_feature:nnnnn {#1} { \00_feat_reset:n {#2} } {} {} {#4}
                                1603
                               1604 }
```

36.1 Adding features when loading fonts

#4 : Tag prefix to activate: +#4 = on, -#4 = off. #5 : Tags to remove in the on case (clist)

When remove clashing features,

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

```
1605 \cs_new:Nn \@@_make_OT_feature:nnn
1606
     {
1607 (debug) \typeout{:: @@_make_OT_feature:nnn \exp_not:n { {#1}{#2}{#3} } }
1608
        \bool_set_true:N \l_@@_proceed_bool
1609
        \bool_set_true:N \l_@@_check_feat_bool
1610
1611
        \tl_if_empty:nT {#1} { \bool_set_false:N \l_@@_check_feat_bool }
1612
        \bool_if:NT \l_@@_check_feat_bool
1613
1614
            \@@_check_ot_feat:nF {#1}
1615
1616
                \@@_warning:nx {icu-feature-not-exist-in-font} {#1}
1617
                \bool_set_false:N \l_@@_proceed_bool
1618
1619
          }
1620
1621
```

```
\bool_if:NT \l_@@_proceed_bool
1622
1623
1624
            \exp_args:Nx \@@_remove_clashing_featstr:n
1625
              { #2 , \@@_swap_plus_minus:n {#2} , #3 }
1626
            \@@_update_featstr:n {#2}
1627
1628
1629
1630 \cs_generate_variant:Nn \00_make_OT_feature:nnn {xxx}
1631\cs_new:Nn \@@_swap_plus_minus:n { \@@_swap_plus_minus_aux:NNNNN #1 }
1632 \cs_new:Nn \@@_swap_plus_minus_aux:NNNNN
    { \t = \{ +\} \{-\#2\#3\#4\#5\} \{-\} \{\#2\#3\#4\#5\} \} \}
```

\@@_check_script:nTF

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

```
1634\prg_new_conditional:Nnn \@@_check_script:n {TF}
1635
        \bool_if:NTF \l_@@_never_check_bool
1636
1637
          { \prg_return_true: }
1638 (*xetexx)
1639 {
1640
     \@@_iv_str_to_num:Nn \l_@@_strnum_int {#1}
     \int set:Nn \l tmpb int { \XeTeXOTcountscripts \l fontspec font }
1641
     \int_zero:N \l_tmpa_int
1642
      \bool_set_false:N \l__fontspec_check_bool
1643
      \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
1644
1645
       \ifnum \XeTeXOTscripttag\l_fontspec_font \l_tmpa_int = \l_@@_strnum_int
1646
          \bool_set_true: N \l__fontspec_check_bool
1647
          \int_set:Nn \l_tmpa_int {\l_tmpb_int}
1648
1649
          \int_incr:N \l_tmpa_int
1650
        \fi
1651
1652
      \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
1653
1654 }
1655 (/xetexx)
1656 (*luatex)
1657 {
     \directlua{fontspec.check_ot_script("l_fontspec_font", "#1")}
     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
1660 }
1661 (/luatex)
```

\@@_check_lang:nTF

This macro takes an OpenType language tag and checks if it exists in the current font/script. The output boolean is $\ensuremath{\texttt{@tempswatrue}}$. $\ensuremath{\texttt{l}}_@@_\ensuremath{\texttt{language}}$ int is used to store the number corresponding to the language tag string. The script used is whatever's held in $\ensuremath{\texttt{l}}_@@_\ensuremath{\texttt{script}}$ int. By default, that's the number corresponding to 'latn'.

```
1663 \prg_new_conditional:Nnn \@@_check_lang:n {TF}
1664
1665
        \bool_if:NTF \l_@@_never_check_bool
1666
          { \prg_return_true: }
1667 (*xetexx)
1668
1669
      \@@_iv_str_to_num:Nn \l_@@_strnum_int {#1}
      \int set:Nn \l tmpb int
1670
       { \XeTeXOTcountlanguages \l_fontspec_font \l_@0_script_int }
1671
      \int_zero:N \l_tmpa_int
1672
      \bool_set_false: N \l__fontspec_check_bool
1673
      \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
1674
1675
        \ifnum\XeTeXOTlanguagetag\l_fontspec_font\l_@@_script_int \l_tmpa_int =\l_@@_strnum_int
1676
          \bool_set_true: N \l__fontspec_check_bool
1677
          \int_set:Nn \l_tmpa_int {\l_tmpb_int}
1678
1679
1680
          \int_incr:N \l_tmpa_int
        \fi
т68т
       }
T682
      \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
1683
1684 }
1685 (/xetexx)
1686 (*luatex)
1687
1688
      \directlua
1689
       {
        fontspec.check ot lang( "l fontspec font", "#1", "\l fontspec script tl" )
1690
1691
      \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
1692
1693 }
1694 (/luatex)
1695
    }
```

\@@_check_ot_feat:nTF

This macro takes an OpenType feature tag and checks if it exists in the current font/script/language. $\lower 2 @_strnum_int$ is used to store the number corresponding to the feature tag string. The script used is whatever's held in $\lower 2 @_script_int$. By default, that's the number corresponding to 'latn'. The language used is $\lower 2 @_language_int$, by default o, the 'default language'.

```
1696 \prg_new_conditional:Nnn \@@_check_ot_feat:n {TF,F}
1697
        \bool_if:NTF \l_@@_never_check_bool
1698
          { \prg_return_true: }
1699
1700 (*xetexx)
1701 {
1702 (debug)\typeout{::~ fontspec check ot feat:n~ {#1}}
      \int set:Nn \l tmpb int
1703
       {
1704
        \XeTeXOTcountfeatures \l_fontspec_font
1705
                                \l_@@_script_int
1706
                                \l_@@_language_int
1707
```

```
}
1708
1709
      \@@_iv_str_to_num:Nn \l_@@_strnum_int {#1}
      \int_zero:N \l_tmpa_int
      \bool_set_false:N \l_@@_check_bool
      \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
1712
1713
        \ifnum\XeTeXOTfeaturetag\l_fontspec_font\l_@0_script_int\l_@0_language_int
1714
             \l_tmpa_int =\l_@@_strnum_int
1715
          \bool_set_true:N \l_@@_check_bool
1716
          \int_set:Nn \l_tmpa_int {\l_tmpb_int}
1717
1718
          \int_incr:N \l_tmpa_int
1719
        \fi
1720
       }
1721
      \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
1722
1723 }
1724 (/xetexx)
1725 (*luatex)
1726 {
1727 (debug)\typeout{::~ fontspec_check_ot_feat:n~ {#1}}
     \directlua
1728
1729
        fontspec.check_ot_feat(
1730
                                 "l_fontspec_font", "#1",
1731
                                 "\l_fontspec_lang_tl", "\l_fontspec_script_tl"
1732
1733
1734
      \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
1735
1736 }
1737 (/luatex)
1738 }
```

36.2 OpenType feature information

```
1739 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {aalt}{Access~All~Alternates}
1740 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvf}{Above-base~Forms}
1741 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvm}{Above-base~Mark~Positioning}
1742 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {abvs}{Above-base~Substitutions}
1743 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {afrc}{Alternative~Fractions}
1744 \prop_gput:\nn \g_@@_all_opentype_feature_names_prop {akhn}{Akhands}
1745 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwf}{Below-base~Forms}
1746\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwm}{Below-base~Mark~Positioning}
1747 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {blws}{Below-base~Substitutions}
1748 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {calt}{Contextual~Alternates}
1749 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {case}{Case-Sensitive~Forms}
1750 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {ccmp}{Glyph~Composition~/~Decomposition
1751 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cfar}{Conjunct~Form~After~Ro}
1752 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cjct}{Conjunct~Forms}
1753 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {clig}{Contextual~Ligatures}
1754 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {cpct}{Centered~CJK~Punctuation}
1755 \prop gput: Nnn \g @@ all opentype feature names prop {cpsp}{Capital~Spacing}
```

```
1756 \prop_gput:Nnn \g_00_all_opentype_feature_names_prop {cswh}{Contextual~Swash}
1757 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {curs}{Cursive~Positioning}
1758 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {cvNN}{Character~Variant~$N$}
1759 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {c2pc}{Petite~Capitals~From~Capitals}
1760 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2sc}{Small~Capitals~From~Capitals}
1761 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {dist}{Distances}
1762 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {dlig}{Discretionary~Ligatures}
1763 \prop_gput:Nnn \g_00_all_opentype_feature_names_prop {dnom}{Denominators}
1764\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dtls}{Dotless~Forms}
1765 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {expt}{Expert~Forms}
1766\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {falt}{Final~Glyph~on~Line~Alternates}
1767\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin2}{Terminal~Forms~\#2}
1768\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin3}{Terminal~Forms~\#3}
1769 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {fina}{Terminal~Forms}
1770 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {flac}{Flattened~accent~forms}
1771 \prop gput: Nnn \g @@ all opentype feature names prop {frac}{Fractions}
1772 \prop gput: Nnn \g @@ all opentype feature names prop {fwid}{Full~Widths}
1773 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {half}{Half~Forms}
1774 \prop gput: Nnn \g @@ all opentype feature names prop {haln}{Halant~Forms}
1775 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {halt}{Alternate~Half~Widths}
1776\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hist}{Historical~Forms}
1777 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hkna}{Horizontal~Kana~Alternates}
1778 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hlig}{Historical~Ligatures}
1779 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {hngl}{Hangul}
1780 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hojo}{Hojo~Kanji~Forms}
1781 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {hwid}{Half~Widths}
1782 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {init}{Initial~Forms}
1783\prop gput:Nnn \g @@ all opentype feature names prop {isol}{Isolated~Forms}
1784 \prop_gput:\nn \g_@@_all_opentype_feature_names_prop {ital}{Italics}
1785 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jalt}{Justification~Alternates}
1786 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp78}{JIS78~Forms}
1787\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp83}{JIS83~Forms}
1788 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jpgo}{JISgo~Forms}
1789 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {jpo4}{JIS2004~Forms}
1790 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {kern}{Kerning}
1791 \prop gput:Nnn \g @@ all opentype feature names prop {lfbd}{Left~Bounds}
1792 \prop gput: Nnn \g @@ all opentype feature names prop {liga}{Standard~Ligatures}
1793 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ljmo}{Leading~Jamo~Forms}
1794 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {lnum}{Lining~Figures}
1795 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {loc1}{Localized~Forms}
1797 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {ltrm}{Left-to-right~mirrored~forms}
1798\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mark}{Mark~Positioning}
1799 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {med2}{Medial~Forms~\#2}
1800 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {medi}{Medial~Forms}
1801 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {mgrk}{Mathematical~Greek}
1802 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mkmk}{Mark~to~Mark~Positioning}
1803 \prop gput: Nnn \g @@ all opentype feature names prop {mset}{Mark~Positioning~via~Substitution}
1804 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {nalt}{Alternate~Annotation~Forms}
1805 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {nlck}{NLC~Kanji~Forms}
```

1806 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nukt}{Nukta~Forms}

```
1807\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {numr}{Numerators}
1808 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {onum}{Oldstyle~Figures}
1809\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {opbd}{Optical~Bounds}
1810 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {ordn}{Ordinals}
1811 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ornm}{Ornaments}
1812 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {palt}{Proportional~Alternate~Widths}
1813 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {pcap}{Petite~Capitals}
1814\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pkna}{Proportional~Kana}
1815 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {pnum}{Proportional~Figures}
1816\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pref}{Pre-Base~Forms}
1817 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pres}{Pre-base~Substitutions}
1818 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstf}{Post-base~Forms}
1819 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {psts}{Post-base~Substitutions}
1820 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {pwid}{Proportional~Widths}
1821 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {qwid}{Quarter~Widths}
1822 \prop gput: Nnn \g @@ all opentype feature names prop {rand}{Randomize}
1823 \prop gput: Nnn \g @@ all opentype feature names prop {rclt}{Required~Contextual~Alternates}
1824 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {rkrf}{Rakar~Forms}
1825 prop gput: Nnn \g @@ all opentype feature names prop {rlig}{Required~Ligatures}
1826\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rphf}{Reph~Forms}
1827 \prop_gput:Nnn \g_00_all_opentype_feature_names_prop {rtbd}{Right~Bounds}
1828 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {rtla}{Right-to-left~alternates}
1829 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {rtlm}{Right-to-left~mirrored~forms}
1830 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {ruby}{Ruby~Notation~Forms}
1831 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {rvrn}{Required~Variation~Alternates}
1832 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {salt}{Stylistic~Alternates}
1833 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {sinf}{Scientific~Inferiors}
1834 \prop gput: Nnn \g @@ all opentype feature names prop {size}{Optical~size}
1835 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smcp}{Small~Capitals}
1836 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {smpl}{Simplified~Forms}
1837 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssNN}{Stylistic~Set~$N$}
1838 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {ssty}{Math~script~style~alternates}
1839 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {stch}{Stretching~Glyph~Decomposition}
1840 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {subs}{Subscript}
1841 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {sups}{Superscript}
1842 \prop gput: Nnn \g @@ all opentype feature names prop {swsh}{Swash}
1843 \prop gput: Nnn \g @@ all opentype feature names prop {titl}{Titling}
1844 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing~Jamo~Forms}
1845 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional~Name~Forms}
1846 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular~Figures}
\label{lem:non_gput:Nnn} $$ 1847 \simeq \sup_{0 \in \mathbb{Z}} \operatorname{const}(0) = \operatorname{const}
1848 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third~Widths}
1849 \prop_gput:\nn \g_00_all_opentype_feature_names_prop {unic}{Unicase}
1850 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate~Vertical~Metrics}
1851 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu~Variants}
1852 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical~Writing}
1853 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate~Vertical~Half~Metrics}
1854\prop gput:Nnn \g @@ all opentype feature names prop {vjmo}{Vowel~Jamo~Forms}
1855 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical~Kana~Alternates}
1856 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkrn}{Vertical~Kerning}
```

1857 \prop_gput: Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional~Alternate~Vertical~Me

37 Graphite/AAT code

```
00 define aat feature group:n
                               1861 \cs_new: Nn \@@_define_aat_feature_group:n
                                    { \keys define:nn {fontspec-aat} { #1 .multichoice: } }
 \@@_define_aat_feature:nnnn
                               1863 \cs_new: Nn \@@_define_aat_feature:nnnn
                                     \keys_define:nn {fontspec-aat}
                               1865
                               1866
                                       #1/#2 .code:n = { \@@_make_AAT_feature:nn {#3}{#4} }
                               1867
                               1868
                               1869
     \@@_make_AAT_feature:nn
                               1870 \cs_new:Nn \@@_make_AAT_feature:nn
                               1871 {
                                    \tl_if_empty:nTF {#1}
                               1872
                                      { \@@_warning:n {aat-feature-not-exist} }
                               1873
                               1874
                                        \@@_make_AAT_feature_string:nnTF {#1}{#2}
                               1875
                               1876
                               1877
                                          \@@_update_featstr:n {\l_fontspec_feature_string_tl}
                               1878
                                        { \@@_warning:nx {aat-feature-not-exist-in-font} {\#1,\#2} }
                               1879
                                      }
                               1880
                               1881 }
```

_make_AAT_feature_string:nnTF

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 108).

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 X₂TEX 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.

```
1882 \prg_new_conditional:Nnn \@@_make_AAT_feature_string:nn {TF,T,F}
1883 {
1884 \tl_set:Nx \l_tmpa_tl { \XeTeXfeaturename \l_fontspec_font #1 }
1885 \tl_if_empty:NTF \l_tmpa_tl
1886 { \prg_return_false: }
1887 {
```

```
\int_compare:nTF { \XeTeXisexclusivefeature\l_fontspec_font #1 > 0 }
1888
1889
1890
          \tl_set:Nx \l_tmpb_tl {\XeTeXselectorname\l_fontspec_font #1\space #2}
1891
1892
         {
1893
          \int_if_even:nTF {#2}
1894
            \tl_set:Nx \l_tmpb_tl {\XeTeXselectorname\l_fontspec_font #1\space #2}
1895
1896
           {
1897
            \tl_set:Nx \l_tmpb_tl
1898
1899
              \XeTeXselectorname\l_fontspec_font #1\space \numexpr#2-1\relax
1901
            \tl_if_empty:NF \l_tmpb_tl { \tl_put_left:Nn \l_tmpb_tl {!} }
1902
1903
1904
        \tl_if_empty:NTF \l_tmpb_tl
1905
         { \prg_return_false: }
1906
1907
          \tl_set:Nx \l_fontspec_feature_string_tl { \l_tmpa_tl = \l_tmpb_tl }
1908
          \prg_return_true:
1909
1910
1911
1912 }
```

38 Font loading (keyval) definitions

1927

This is the tedious section where we correlate all possible (eventually) font feature requests with their X¬TEX representations.

```
1913 \clist_set:Nn \g_@@_all_keyval_modules_clist
1914
        fontspec, fontspec-opentype, fontspec-aat,
1915
        fontspec-preparse, fontspec-preparse-external, fontspec-preparse-nested,
1916
        fontspec-renderer
1917
1918
1919 \cs_new: Nn \@@_keys_define_code:nnn
1920
       \ensuremath{\mbox{keys\_define:nn } \{\#1\} \ \{ \#2 \ .code:n = \{\#3\} \ \}}
1921
1922
     For catching features that cannot be used in \addfontfeatures:
1923 \cs_new:Nn \@@_aff_error:n
1924
        \@@_keys_define_code:nnn {fontspec-addfeatures} {#1}
1925
1926
          { \@@_error:nx {not-in-addfontfeatures} {#1} }
```

38.0.1 Pre-parsing naming information

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

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

```
1928 \@@_keys_define_code:nnn {fontspec-preparse-external} {Path}
1929 {
1930 \bool_set_true:N \l_@@_nobf_bool
1931 \bool_set_true:N \l_@@_noit_bool
1932 \bool_set_true:N \l_@@_external_bool
1933 \tl_set:Nn \l_@@_font_path_tl {#1}
1934 \@@_font_is_file:
1935 \extexx
1936 \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
1937 \(/xetexx\)
1938 }
1939 \aliasfontfeature{Path}{ExternalLocation}
1940 \@@_keys_define_code:nnn {fontspec} {Path} {}
```

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

```
1941 \@@_keys_define_code:nnn {fontspec-preparse-external} {Extension}
1942 {
1943  \tl_set:Nn \l_@@_extension_tl {#1}
1944  \bool_if:NF \l_@@_external_bool
1945  {
1946   \keys_set:nn {fontspec-preparse-external} {Path}
1947  }
1948 }
1949 \tl_clear:N \l_@@_extension_tl
1950 \@@_keys_define_code:nnn {fontspec} {Extension} {}
```

38.0.2 Pre-parsed features

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

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

```
1951 \keys_define:nn {fontspec-renderer}
1952 {
1953    Renderer .choices:nn =
1954    {AAT,ICU,OpenType,Graphite,Full,Basic}
1955    {
1956    \int_compare:nTF {\l_keys_choice_int <= 4} {
1957 \*xetexx\}
1958    \t1_set:Nv \l_fontspec_renderer_tl</pre>
```

```
{ g_fontspec_renderer_tag_ \l_keys_choice_tl }
1959
          \tl_gset:Nx \g_00_single_feat_tl { \l_fontspec_renderer_tl }
1960
1961 (/xetexx)
1962 (*luatex)
1963
          \@@_warning:nx {only-xetex-feature} {Renderer=AAT/OpenType/Graphite}
1964 (/luatex)
1965
1966
         {
1967 (*xetexx)
          \@@ warning:nx {only-luatex-feature} {Renderer=Full/Basic}
1969 (/xetexx)
1970 (*luatex)
          \tl_set:Nv \l_fontspec_mode_tl
1971
            { g_fontspec_mode_tag_ \l_keys_choice_tl }
          \tl_gset:Nx \g_@@_single_feat_tl { mode=\l_fontspec_mode_tl }
1973
1974 (/luatex)
1975
         }
1976
1977 }
1978 \tl_set:cn {g_fontspec_renderer_tag_AAT} {/AAT}
1979 \tl_set:cn {g_fontspec_renderer_tag_ICU} {/OT}
1980 \tl_set:cn {g_fontspec_renderer_tag_OpenType} {/OT}
1981 \tl_set:cn {g_fontspec_renderer_tag_Graphite} {/GR}
1982 \tl_set:cn {g_fontspec_mode_tag_Full} {node}
1983 \tl_set:cn {g_fontspec_mode_tag_Basic} {base}
```

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

```
1984 \@@_keys_define_code:nnn {fontspec-preparse} {Script}
1985 {
              \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
1986 (xetexx)
1987 \tl_set:Nn \l_@0_script_name_tl {#1}
1988 }
```

Exactly the same:

```
1989 \@@_keys_define_code:nnn {fontspec-preparse} {Language}
1991 (xetexx)
              \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
1992 \tl_set:Nn \l_@@_lang_name_tl {#1}
1993 }
```

TTC font index

```
1994 \@@_keys_define_code:nnn {fontspec-preparse} {FontIndex}
1995 {
     \str_if_eq_x:nnF { \str_lower_case:f {\l_@0_extension_tl} } {.ttc}
1996
       { \@@_warning:n {font-index-needs-ttc} }
1998 (xetexx) \tl_set:Nn \l_@@_ttc_index_tl {:#1}
1999 (luatex) \tl set:Nn \l @@ ttc index tl {(#1)}
2001 \00 keys define code:nnn {fontspec} {FontIndex}
2002 {
```

38.0.3 Bold/italic choosing options

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

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

```
2006 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSeries}
2007 {
              \tl gset:Nx \g @@ curr series tl { #1 }
              \seq gput right: Nx \g @@ bf series seq { #1 }
2010 }
   Fonts Upright:
2011 \00_keys_define_code:nnn {fontspec-preparse-external} {UprightFont}
               \fontspec complete fontname: Nn \l @@ fontname up tl {#1}
2014 }
2015 \00 keys define code:nnn {fontspec-preparse-external} {FontName}
2016
               \fontspec_complete_fontname: Nn \l_@@_fontname_up_tl {#1}
2017
2018 }
   Bold:
2019 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldFont}
2020 {
               \tl if empty:nTF {#1}
2021
                  {
2022
                     \bool_set_true:N \l_@@_nobf_bool
2023
2024
2025
                     \bool_set_false:N \l_@@_nobf_bool
2026
                     \fontspec_complete_fontname: Nn \l_@@_curr_bfname_tl {#1}
2027
2028
                     \seq_if_empty:NT \g_@@_bf_series_seq
2029
2030
                          \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
2031
                          \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}} \ensuremath{$\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{}} \ensuremath{$\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{$}\ensuremath{}\ensuremath{$}\ensuremath{$}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\ensuremath{}\en
2032
2033
                     \tl_if_eq:oxT \g_00_curr_series_tl {\bfdefault}
2034
                       { \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl }
2035
2036
2037 (debug)\typeout{Setting~bold~font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
2038
                     \prop put:NxV \l @@ nfss prop
2039
                        {BoldFont-\g @@ curr series tl} \l @@ curr bfname tl
2040
2041
```

```
}
2042
2043 }
 Same for italic:
2044 \00 keys define code:nnn {fontspec-preparse-external} {ItalicFont}
2045 {
     \tl_if_empty:nTF {#1}
2046
       {
2047
        \bool_set_true:N \l_@@_noit_bool
2048
2049
2050
        \bool_set_false:N \l_@@_noit_bool
2051
        \fontspec_complete_fontname: Nn \l_@@_fontname_it_tl {#1}
2052
2053
2054 }
 Simpler for bold+italic & slanted:
2055 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldItalicFont}
2056 {
      \fontspec_complete_fontname: Nn \l_@@_fontname_bfit_tl {#1}
2057
2058
2059 \@@_keys_define_code:nnn {fontspec-preparse-external} {SlantedFont}
2060 {
      \fontspec_complete_fontname: Nn \l_@@_fontname_sl_tl {#1}
2061
2062
2063 \@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSlantedFont}
      \fontspec complete fontname: Nn \l @@ fontname bfsl tl {#1}
2065
2066
 Small caps isn't pre-parsed because it can vary with others above:
2067 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
2068 {
     \tl if empty:nTF {#1}
2069
       {
2070
        \bool_set_true:N \l_@@_nosc_bool
2071
2072
       {
2073
        \bool_set_false:N \l_@@_nosc_bool
2074
        \fontspec_complete_fontname: Nn \l_@@_fontname_sc_tl {#1}
2076
2077 }
 Features
2078 \@@_keys_define_code:nnn {fontspec-preparse} {UprightFeatures}
2079 {
     \clist_set:Nn \l_@@_fontfeat_up_clist {#1}
2080
2081
2082 \@@_keys_define_code:nnn {fontspec-preparse} {BoldFeatures}
2083 {
     \clist_set:Nn \l_@@_fontfeat_bf_clist {#1}
2084
2085
```

```
\prop_put:NxV \l_@@_nfss_prop
2086 %
          {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
2087 %
2088 }
2089 \@@_keys_define_code:nnn {fontspec-preparse} {ItalicFeatures}
2090 {
2091
      \clist_set:Nn \l_@@_fontfeat_it_clist {#1}
2092 }
2093 \00_keys_define_code:nnn {fontspec-preparse} {BoldItalicFeatures}
2094 {
      \clist set:Nn \l @@ fontfeat bfit clist {#1}
2095
2096 }
2097 \@@_keys_define_code:nnn {fontspec-preparse} {SlantedFeatures}
      \clist_set:Nn \l_@@_fontfeat_sl_clist {#1}
2101 \@@ keys define code:nnn {fontspec-preparse} {BoldSlantedFeatures}
2102 {
      \clist_set:Nn \l_@@_fontfeat_bfsl_clist {#1}
2103
2104 }
 Note that small caps features can vary by shape, so these in fact aren't pre-parsed.
2105 \000_keys_define_code:nnn {fontspec} {SmallCapsFeatures}
2106 {
      \bool_if:NF \l_@@_firsttime_bool
2107
2108
        \clist_set:Nn \l_@@_fontfeat_sc_clist {#1}
2110
     paragraphFeatures varying by size
2112 \@@_keys_define_code:nnn {fontspec-preparse} {SizeFeatures}
2113 {
     \clist set:Nn \l @@ sizefeat clist {#1}
     \clist_put_right:Nn \l_@0_fontfeat_up_clist { SizeFeatures = {#1} }
2116 }
2117 \@@_keys_define_code:nnn {fontspec-preparse-nested} {SizeFeatures}
      \clist_set:Nn \l_@@_sizefeat_clist {#1}
     \tl_if_empty:NT \l_@@_this_font_tl
       \{ tl_set: Nn \l_00_this_font_tl \{ -- \} \} \% needs to be non-empty as a flag
2123 \@@_keys_define_code:nnn {fontspec-preparse-nested} {Font}
2124 {
     \t: Nn \l_00_this_font_tl {#1}
2125
2127 \@@_keys_define_code:nnn {fontspec} {SizeFeatures}
2128 {
     % dummy
2129
2131 \00 keys define code:nnn {fontspec} {Font}
2132 {
2133
     % dummy
2134 }
```

```
2135 \@@_keys_define_code:nnn {fontspec-sizing} {Size}
2136 {
2137  \tl_set:Nn \l_@@_size_tl {#1}
2138 }
2139 \@@_keys_define_code:nnn {fontspec-sizing} {Font}
2140 {
2141  \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
2142 }
```

38.0.4 Font-independent features

These features can be applied to any font.

NFSS encoding For the very brave.

```
2143 \@@_keys_define_code:nnn {fontspec-preparse} {NFSSEncoding}
2144 {
2145 \tl_gset:Nx \l_@@_nfss_enc_tl { #1 }
2146 }
```

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.)

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

```
2154 \@@_keys_define_code:nnn {fontspec} {FontFace}
2155 {
     \tl_set:No \l_@@_arg_tl { \use_iii:nnn #1 }
2156
     \tl_set_eq:NN \l_@@_this_feat_tl \l_@@_arg_tl
     \tl_clear:N \l_@@_this_font_tl
2159 \int_compare:nT { \clist_count:N \l_@@_arg_tl = 1 }
2160
2161 (*debug)
        \typeout{FontFace~ parsing:~ one~ clist~ item}
2162
2163 (/debug)
       \tl_if_in:NnF \l_@@_arg_tl {=}
2164
2165
        {
2166 (*debug)
          \typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
2168 (/debug)
          \tl_set_eq:NN \l_@0_this_font_tl \l_@0_arg_tl
2169
          \tl clear:N \l @@ this feat tl
2170
```

```
2171     }
2172     }
2173
2174     \@@_add_nfssfont:nnnn
2175      {\use_i:nnn #1}{\\l_@@_this_font_tl}{\\l_@@_this_feat_tl}}
2176 }
```

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.

\@@_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¬T¬x).

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

```
2187 \cs_new:Nn \@@_calc_scale:n
2188 {
2189
                                                                      \group begin:
                                                                                                   \rmfamily
2190
                                                                                                 \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
2191
                                                                                                 \label{local_set_font_dimen:NnN l_00_tmpb_dim {#1} \l_fontspec_font} $$ \end{substitute} % $$ \column{2.5cm} $$ \colum
2192
                                                                                                 \tl_gset:Nx \l_@0_scale_tl
2193
2194
                                                                                                                         \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} / \dim_to_fp:n {\
                                                                                                                                                                                                                                                                                             \dim_to_fp:n {\l_@@_tmpb_dim} }
2196
2197
                                                                                                 \00 info:n {set-scale}
2198
                                                                      \group_end:
2199
2200 }
```

\@@_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'.

```
2201\cs_new:Nn \@@_set_font_dimen:NnN
2202 {
2203 \dim_set:Nn #1 { \fontdimen #2 #3 }
2204 \dim_compare:nNnT #1 = {opt}
```

```
2205
2206
        \settoheight #1
2207
2208
           \str_if_eq:nnTF {#_3} {\font} \rmfamily #_3
2209
           \int_case:nnF #2
2210
              \{5\} \{x\} % x-height
2211
              \{8\} \{X\} % cap-height
2212
            } {?} % "else" clause; never reached.
2213
2214
       }
2215
2216 }
```

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

```
2217 \@@_keys_define_code:nnn {fontspec} {WordSpace}
2218 {
2219 \bool_if:NF \l_@@_firsttime_bool
2220 { \_fontspec_parse_wordspace:w #1,,,\q_stop }
2221 }
2222 \@@_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 $\{X,X,X\}$.

```
2223 \cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
2224 {
      \tl_if_empty:nTF {#4}
2225
2226
       {
        \tl_set:Nn \l_@@_wordspace_adjust_tl
2227
         {
2228
          \fontdimen 2 \font = #1 \fontdimen 2 \font
2229
          \fontdimen 3 \font = #1 \fontdimen 3 \font
2230
          \fontdimen 4 \font = #1 \fontdimen 4 \font
2231
2232
2233
       }
       {
2234
        \tl_set:Nn \l_@@_wordspace_adjust_tl
2235
2236
          \fontdimen 2 \font = #1 \fontdimen 2 \font
2237
          \fontdimen 3 \font = #2 \fontdimen 3 \font
2238
          \fontdimen 4 \font = #3 \fontdimen 4 \font
2239
2240
       }
2241
2242 }
```

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

```
2243 \@@_keys_define_code:nnn {fontspec} {PunctuationSpace}
2244 {
2245 \str_case_x:nnF {#1}
2246 {
```

```
{WordSpace}
2247
2248
2249
         \tl_set:Nn \l_@@_punctspace_adjust_tl
          { \fontdimen 7 \font = 0 \fontdimen 2 \font }
2250
2251
        {TwiceWordSpace}
2252
2253
         \tl_set:Nn \l_@@_punctspace_adjust_tl
2254
          { \fontdimen 7 \font = 1 \fontdimen 2 \font }
2255
2256
       }
2257
2258
         \tl_set:Nn \l_@@_punctspace_adjust_tl
2259
         { \fontdimen 7 \font = #1 \fontdimen 7 \font }
2260
2261
2262 }
2263 \@@_aff_error:n {PunctuationSpace}
```

Secret hook into the font-adjustment code

```
2264 \@@_keys_define_code:nnn {fontspec} {FontAdjustment}
2265 {
2266 \tl_put_right:Nx \l_@@_postadjust_tl {#1}
2267 }
```

Letterspacing

```
2268 \@@_keys_define_code:nnn {fontspec} {LetterSpace}
2269 {
2270 \@@_update_featstr:n {letterspace=#1}
2271 }
```

Hyphenation character This feature takes one of three arguments: 'None', $\langle glyph \rangle$, or $\langle slot \rangle$. If the input isn't the first, and it's one character, then it's the second; otherwise, it's the third.

```
2272 \00_keys_define_code:nnn {fontspec} {HyphenChar}
     \bool_if:NT \l_@@_addfontfeatures_bool
2274
      { \@@_error:nx {not-in-addfontfeatures} {HyphenChar} }
2275
2276
      \str_if_eq:nnTF {#1} {None}
2277
2278
       \tl_put_right:Nn \l_@@_postadjust_tl
2279
          { \hyphenchar \font = \c_minus_one }
2280
2281
2282
        \tl_if_single:nTF {#1}
2283
        { \tl_set:Nn \l_fontspec_hyphenchar_tl {`#1} }
2284
         { \tl set:Nn \l fontspec hyphenchar tl { #1} }
2285
        \@@_primitive_font_glyph_if_exist:NnTF \l_fontspec_font {\l_fontspec_hyphenchar_tl}
2286
2287
2288
         \tl_put_right:Nn \l_@@_postadjust_tl
```

```
2289 (*xetexx)
           { \hyphenchar \font = \l_fontspec_hyphenchar_tl \scan_stop: }
2291 (/xetexx)
2292 (*luatex)
2293
              \mbox{hyphenchar }\mbox{font = \c_zero}
2294
              \int_set:Nn \luatex_prehyphenchar:D { \l_fontspec_hyphenchar_tl }
2295
2296
2297 (/luatex)
2298
        { \color{00\_error:nx {no-glyph}{#1} } }
2299
      }
2300
2301 }
2302 \@@_aff_error:n {HyphenChar}
 Color Hooks into pkgxcolor, which names its colours \color@<name>.
2303 \@@_keys_define_code:nnn {fontspec} {Color}
2304 {
     \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
2305
2306
       \convertcolorspec{named}{#1}{HTML}\l @@ hexcol tl
2307
2308
2309
       2310
        { \tl_set:Nn \l_@@_hexcol_tl {#1} }
2311
2312
         2313
2314
           { \fontspec_parse_colour:viii #1 }
2315
           \bool_if:NF \l_@@_firsttime_bool
2316
            { \@@_warning:nx {bad-colour} {#1} }
2317
2318
        }
2319
      }
2320
2321 }
2322 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
2323 {
     \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
2324
     \tl_if_eq:NNF \l_@@_opacity_tl \g_@@_opacity_tl
2325
2326
       \bool_if:NF \l_@@_firsttime_bool
2327
        { \@@_warning:nx {opa-twice-col} {#7#8} }
2328
2329
     \tl_set:Nn \l_@@_opacity_tl {#7#8}
2330
2331 }
2332 \aliasfontfeature{Color}{Colour}
2333 \@@_keys_define_code:nnn {fontspec} {Opacity}
2334 {
     \int_set:Nn \l_@@_tmp_int {255}
2335
     \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
```

```
\tl_if_eq:NNF \l_@0_opacity_tl \g_@0_opacity_tl
2337
2338
        \bool_if:NF \l_@@_firsttime_bool
2339
         { \00_{\text{warning:nx}} \{ \text{opa-twice} \} 
2340
2341
      \tl_set:Nx \l_@@_opacity_tl
2342
2343
         \label{local_compare:nT { l_00_tmp_int <= "F } {o} % zero pad}
2344
         \int_to_hex:n { \l_@@_tmp_int }
2345
       }
2346
2347 }
```

Mapping

```
2348 \@@_keys_define_code:nnn {fontspec} {Mapping}
<sub>2349</sub> (*xetexx)
2350 {
     \@@_update_featstr:n { mapping = #1 }
2351
2352 }
2353 (/xetexx)
2354 (*luatex)
2355 {
2356
      \str_if_eq:nnTF {#1} {tex-text}
2357
        \@@_warning:n {no-mapping-ligtex}
2358
        \msg_redirect_name:nnn {fontspec} {no-mapping-ligtex} {none}
2359
        \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
2360
2361
       { \@@_warning:n {no-mapping} }
2362
2363 }
2364 (/luatex)
```

38.0.5 Continuous font axes

```
2365 \@@_keys_define_code:nnn {fontspec} {Weight}
2366 {
     \@@_update_featstr:n{weight=#1}
2367
2368 }
2369 \@@_keys_define_code:nnn {fontspec} {Width}
2370 {
     \@@_update_featstr:n{width=#1}
2371
2372 }
2373 \00_keys_define_code:nnn {fontspec} {OpticalSize}
2374 (*xetexx)
2375 {
     \bool_if:NTF \l_@@_ot_bool
2376
2377
       \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
2378
      }
2379
2380
2381
        \bool_if:NT \l_@@_mm_bool
2382
         {
```

```
\@@_update_featstr:n { optical size = #1 }
2383
         }
2384
2385
      \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
2386
2387
        \bool_if:NT \l_@@_firsttime_bool
2388
2389
         { \@@_warning:n {no-opticals} }
2390
2391 }
2392 (/xetexx)
2393 (*luatex)
2394 {
     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
2396 }
2397 (/luatex)
```

38.0.6 Font transformations

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

```
2398 \keys_define:nn {fontspec}
2399 {
     FakeSlant .code:n =
2400
2401
        \verb|\@@_update_featstr:n{slant=\#_1}|
2402
       },
2403
      FakeSlant .default:n = {0.2}
2404
2405 }
2406 \keys_define:nn {fontspec}
2407 {
2408
     FakeStretch .code:n =
2409
        \@@_update_featstr:n{extend=#1}
2410
       },
2411
      FakeStretch .default:n = {1.2}
2412
2413 }
2414 (*xetexx)
2415 \keys_define:nn {fontspec}
2416 {
      FakeBold.code:n =
2417
2418
        \@@ update featstr:n {embolden=#1}
2419
       },
2420
     FakeBold .default:n = \{1.5\}
2421
2422 }
2423 (/xetexx)
2424 (*luatex)
2425 \keys_define:nn {fontspec}
     FakeBold .code:n = { \@@_warning:n {fakebold-only-xetex} }
2427
2428 }
<sub>2429</sub> (/luatex)
```

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.)

```
2430 \keys_define:nn {fontspec}
2431 {
     AutoFakeSlant .code:n =
2432
2433
        \bool_if:NT \l_@@_firsttime_bool
2434
2435
          \tl_set:Nn \l_@0_fake_slant_tl {#1}
2436
          \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
2437
2438
          \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
          \bool set false: N \l @@ noit bool
2439
2440
          \tl_if_empty:NF \l_@@_fake_embolden_tl
244I
2442
            \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
2443
             {\tt FakeBold=\l_@0\_fake\_embolden\_tl}\\
2444
            \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
2445
            \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
2446
2447
         }
2448
       },
2449
      AutoFakeSlant .default:n = {0.2}
2450
2451}
 Same but reversed:
2452 \keys_define:nn {fontspec}
2453 {
     AutoFakeBold .code:n =
2454
2455
        \bool_if:NT \l_@@_firsttime_bool
2456
2457
          \label{local_to_model} $$ \tilde{\ }_{\rm embolden_tl} $$
2458
          \clist_put_right:\n\\l_@@_fontfeat_bf_clist {FakeBold=#1}
2459
          \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
2460
          \bool_set_false: N \l_@@_nobf_bool
2461
2462
          \tl if empty:NF \l @@ fake slant tl
2463
2464
            \clist put right:Nx \l @@ fontfeat bfit clist
2465
             {FakeSlant=\l @@ fake slant tl}
2466
            \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
2467
            \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
2468
2469
         }
2470
       },
2471
      AutoFakeBold .default:n = \{1.5\}
```

38.0.7 Raw feature string

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

```
2474 \@@_keys_define_code:nnn {fontspec} {RawFeature}
     \@@_update_featstr:n {#1}
2476
2477 }
```

OpenType feature definitions

```
2478 \@@_feat_prop_add:nn {salt} { Alternate\,=\,$N$ }
2479\00_feat_prop_add:nn {nalt} { Annotation\,=\,$N$ }
2480\00_{\text{feat\_prop\_add:nn }} \{ \text{Ornament}, =\,\$N\$ \}
2481 \@@_feat_prop_add:nn {cvNN} { CharacterVariant\,=\,$N$:$M$ }
2482 \@@_feat_prop_add:nn {ssNN} { StylisticSet\,=\,$N$ }
```

38.2 Regular key=val / tag definitions

38.2.1 Ligatures

```
2483 \@@_define_opentype_feature_group:n {Ligatures}
2484 \@@_define_opentype_feature:nnnnn {Ligatures} {ResetAll} {} {}
2485
       +dlig,-dlig,+rlig,-rlig,+liga,-liga,+dlig,-dlig,+clig,-clig,+hlig,-hlig,
2487 (xetexx) mapping = tex-text
2488 (luatex) +tlig,-tlig
2489
2490 \@@ define opentype onoffreset:nnnnn {Ligatures} {Required}
                                                                       {rlig} {rlig} {}
2491 \@@ define opentype onoffreset:nnnnn {Ligatures} {Common}
                                                                       {liga} {liga} {}
2492 \00 define opentype onoffreset:nnnnn {Ligatures} {Rare}
                                                                       {dlig} {dlig} {}
2493 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Discretionary} {dlig} {dlig} {}
2494 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Contextual}
                                                                       {clig} {clig} {}
2495 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Historic}
                                                                       {hlig} {hlig} {}
```

Emulate CM extra ligatures.

```
2496 (xetexx)\00_define_opentype_onreset:nnnnn {Ligatures} {TeX} {} { mapping = tex-
   text } {}
2497 (luatex)\@@ define opentype onreset:nnnnn {Ligatures} {TeX} {} { +tlig } {}
```

38.2.2 Letters

```
2498 \@@_define_opentype_feature_group:n {Letters}
2499 \@@ define opentype feature:nnnnn
                                        {Letters} {ResetAll} {} {}
2500
        +case, +smcp, +pcap, +c2sc, +c2pc, +unic, +rand,
2501
2502
        -case,-smcp,-pcap,-c2sc,-c2pc,-unic,-rand
     }
2503
```

```
 2504 \\ \ensuremath{\texttt{00\_define\_opentype\_onoffreset:nnnnn}} $$ \{Uppercase\} $$ \{case\} $$ \{-smcp, +pcap, +c2sc, +c
2505 \@@_define_opentype_onoffreset:nnnnn {Letters} {SmallCaps} {smcp} {+pcap,+unic,+rand}
2506 \@@ define opentype onoffreset:nnnnn {Letters} {PetiteCaps} {pcap} {pcap} {+smcp,+unic,+rand}
```

```
2509 \@@_define_opentype_onoffreset:nnnnn {Letters} {Unicase} {unic} {+rand}
2510 \@@_define_opentype_onoffreset:nnnnn {Letters} {Random} {rand} {+unic}
 38.2.3 Numbers
2511 \@@_define_opentype_feature_group:n {Numbers}
2512 \@@_define_opentype_feature:nnnnn {Numbers} {ResetAll} {} {}
2513
       +tnum,-tnum,
2514
       +pnum,-pnum,
2515
2516
       +onum, -onum,
       +lnum,-lnum,
2517
       +zero.-zero.
2518
       +anum, -anum,
2519
2520 }
2521 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Monospaced}
                                                                 {tnum} {tnum} {+pnum,-
   pnum}
2522 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Proportional} {pnum} {+tnum,-
   tnum}
2523 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Lowercase}
                                                                 {onum} {onum} {+lnum,-
   lnum}
2524 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Uppercase}
                                                                 {lnum} {lnum} {+onum,-
2525 \@@ define opentype onoffreset:nnnnn {Numbers} {SlashedZero}
                                                                {zero} {zero} {}
2526 \aliasfontfeatureoption {Numbers} {Lowercase} {OldStyle}
2527 \aliasfontfeatureoption {Numbers} {Uppercase} {Lining}
    luaotload provides a custom anum feature for replacing Latin (AKA Arabic) num-
 bers with Arabic (AKA Indic-Arabic). The same feature maps to Farsi (Persian) num-
 bers if font language is Farsi.
2528 (luatex) \@@_define_opentype_onoffreset:nnnnn {Numbers} {Arabic} {anum} {}}
 38.2.4 Vertical position
2529 \00_define_opentype_feature_group:n {VerticalPosition}
_{2530}\ @@_define_opentype_feature:nnnnn
                                        {VerticalPosition} {ResetAll} {} {}
2531
       +sups,-sups,
2532
       +subs,-subs,
2533
       +ordn,-ordn,
2534
       +numr,-numr,
2535
       +dnom,-dnom,
2536
       +sinf,-sinf,
2537
     }
2538
2539 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Superior}
                                                                               {sups} {sups} {+
2540 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Inferior}
                                                                                {subs} {subs} {+
2541 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Ordinal}
                                                                               {ordn} {ordn} {+
2542 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Numerator}
                                                                               {numr} {numr} {+
2543 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Denominator}
                                                                               {dnom} {dnom} {+
2544 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {ScientificInferior} {sinf} {sinf} {+:
```

2507 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercaseSmallCaps} {c2sc} {c2sc} {+c2pc,+unic

38.2.5 Contextuals

```
2545 \00 define opentype feature group:n {Contextuals}
2546 \@@ define opentype feature:nnnnn
                                          {Contextuals} {ResetAll} {} {}
2547
       +cswh,-cswh,
2548
       +calt,-calt,
2549
       +init,-init,
2550
       +fina,-fina,
       +falt,-falt,
2553
       +medi,-medi,
2554
     }
{\tt 2555} \verb|\@O_define_opentype_onoffreset:nnnnn} \ \{Contextuals\} \ \{Swash\}
                                                                       {cswh} {cswh} {}
2556 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Alternate}
                                                                       {calt} {calt} {}
2557 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordInitial} {init} {}
2558 \@@ define opentype onoffreset:nnnnn {Contextuals} {WordFinal}
                                                                       {fina} {fina} {}
2559 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {LineFinal}
                                                                       {falt} {falt} {}
2560 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Inner}
                                                                       {medi} {medi} {}
 38.2.6 Diacritics
2561 \@@_define_opentype_feature_group:n {Diacritics}
2562 \@@_define_opentype_feature:nnnnn
                                          {Diacritics} {ResetAll} {} {}
2563
    {
       +mark,-mark,
2564
       +mkmk,-mkmk,
2565
       +abvm,-abvm,
2566
       +blwm,-blwm,
2567
2568
2569 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToBase} {mark} {mark} {}
2570 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToMark} {mkmk} {mkmk} {}
2571 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {AboveBase} {abvm} {abvm} {}
2572 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {BelowBase} {blwm} {blwm} {}
 38.2.7 Kerning
2573 \@@_define_opentype_feature_group:n {Kerning}
_{2574}\@0_{define\_opentype\_feature:nnnnn}
                                          {Kerning} {ResetAll} {} {}
2575
       +cpsp,-cpsp,
2576
       +kern,-kern,
2577
2578
2579 \@@_define_opentype_onoffreset:nnnnn {Kerning} {Uppercase} {cpsp} {}
2580 \@@_define_opentype_feature:nnnnn
                                          {Kerning} {On}
                                                                 {kern} {+kern} {-
2581\00_define_opentype_feature:nnnnn
                                                                 {kern} {-kern} {+kern}
                                          {Kerning} {Off}
                                          {Kerning} {Reset}
                                                                 {} {} {+kern,-kern}
2582 \@@_define_opentype_feature:nnnnn
 38.2.8 Fractions
2583 \00_define_opentype_feature_group:n {Fractions}
2584 \@@_define_opentype_feature:nnnnn
                                          {Fractions} {ResetAll} {} {}
2585 {
       +frac,-frac,
2586
```

```
2587
       +afrc,-afrc,
     }
2588
                                          {Fractions} {On}
                                                               {frac} {+frac} {}
2589 \@@_define_opentype_feature:nnnnn
                                                               {frac} {-frac} {}
2590 \@@_define_opentype_feature:nnnnn
                                          {Fractions} {Off}
                                          {Fractions} {Reset} {} {} {+frac,-frac}
2591 \@@_define_opentype_feature:nnnnn
2592 \@@_define_opentype_onoffreset:nnnnn {Fractions} {Alternate} {afrc} {afrc} {-
   frac}
 38.2.9 Style
2593 \@@_define_opentype_feature_group:n {Style}
                                          {Style} {ResetAll} {} {}
2594 \@@_define_opentype_feature:nnnnn
2595
        +salt,-salt,
2596
       +ital,-ital,
2597
       +ruby,-ruby,
2598
       +swsh,-swsh,
2599
       +hist,-hist,
2600
       +titl,-titl,
2601
       +hkna,-hkna,
2602
       +vkna,-vkna,
2603
        +ssty=o,-ssty=o,
2605
        +ssty=1,-ssty=1,
2607 \@@_define_opentype_onoffreset:nnnnn {Style} {Alternate}
                                                                      {salt} {salt} {}
2608 \@@_define_opentype_onoffreset:nnnnn {Style} {Italic}
                                                                      {ital} {ital} {}
2609 \@@_define_opentype_onoffreset:nnnnn {Style} {Ruby}
                                                                      {ruby} {ruby} {}
2610 \@@_define_opentype_onoffreset:nnnnn {Style} {Swash}
                                                                      {swsh} {swsh} {}
2611 \00 define opentype onoffreset:nnnnn {Style} {Cursive}
                                                                      {swsh} {curs} {}
2612 \@@_define_opentype_onoffreset:nnnnn {Style} {Historic}
                                                                      {hist} {hist} {}
2613 \@@_define_opentype_onoffreset:nnnnn {Style} {TitlingCaps}
                                                                      {titl} {titl} {}
2614 \@@_define_opentype_onoffreset:nnnnn {Style} {HorizontalKana}
                                                                      {hkna} {hkna} {+vkna,+pkna}
2615 \00_define_opentype_onoffreset:nnnnn {Style} {VerticalKana}
                                                                      {vkna} {vkna} {+hkna,+pkna}
2616 \@@_define_opentype_onoffreset:nnnnn {Style} {ProportionalKana} {pkna} {+vkna,+hkna}
2617 \@@_define_opentype_feature:nnnnn
                                          {Style} {MathScript}
                                                                      \{ssty\} \{+ssty=0\} \{+ssty=1\}
2618 \@@_define_opentype_feature:nnnnn
                                          {Style} {MathScriptScript} {ssty} {+ssty=1} {+ssty=0}
 38.2.10 CJK shape
2619 \@@_define_opentype_feature_group:n {CJKShape}
2620 \@@_define_opentype_feature:nnnnn
                                          {CJKShape} {ResetAll} {} {}
2621
        +trad,-trad,
2622
2623
       +smpl,-smpl,
2624
       +jp78,-jp78,
       +jp83,-jp83,
2625
2626
       +jpgo,-jpgo,
2627
       +jpo4,-jpo4,
2628
       +expt,-expt,
       +nlck,-nlck,
2629
    }
2630
2631 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Traditional} {trad} {+smpl,+jp78,+jp8
```

```
{\tt 2632 \setminus @@\_define\_opentype\_onoffreset:nnnnn} \  \  \{ CJKShape \} \  \  \{ Simplified \}
                                                                       {smpl} {smpl} {+trad,+jp78,+jp8g
                                                                       {jp78} {jp78} {+trad,+smpl,+jp8g
2633 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978}
2634 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983}
                                                                       {jp83} {jp83} {+trad,+smpl,+jp78
2635 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990}
                                                                       {jpgo} {jpgo} {+trad,+smpl,+jp78
2636 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS2004}
                                                                       {jpo4} {jpo4} {+trad,+smpl,+jp78
2637 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Expert}
                                                                       {expt} {expt} {+trad,+smpl,+jp78
2638 \@@_define_opentype_onoffreset:nnnnn {CJKShape} {NLC}
                                                                       {nlck} {nlck} {+trad,+smpl,+jp78
 38.2.11 Character width
2639 \@@_define_opentype_feature_group:n {CharacterWidth}
```

```
2640 \@@_define_opentype_feature:nnnnn
                                         {CharacterWidth} {ResetAll} {} {}
```

2650 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Proportional}

2651 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Full}

```
+fwid,-fwid,
2643
        +hwid,-hwid,
2644
        +twid,-twid,
2645
        +qwid,-qwid,
2646
        +palt,-palt,
2647
2648
        +halt,-halt,
2649
```

+pwid,-pwid,

2641

2642

```
2652 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Half}
                                                                                     {hwid} {hwid} {-
2653 \@@ define opentype onoffreset:nnnnn {CharacterWidth} {Third}
                                                                                     {twid} {twid} {-
2654 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Quarter}
                                                                                     {qwid} {qwid} {-
2655 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateProportional} {palt} {palt} {-
                                                                                    {halt} {halt} {-
2656 \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateHalf}
```

{pwid} {pwid} {-

{fwid} {fwid} {-

38.2.12 Vertical

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

```
2657 \@@_define_opentype_feature_group:n {Vertical}
2658 \@@_define_opentype_onoffreset:nnnnn {Vertical} {RotatedGlyphs}
                                                                             {vrt2} {vrt2} {+vrtr,
2659 \@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternatesForRotation} {vrtr} {vrtr} {+vrt2}
                                                                             {vert} {vert} {+vrt2}
2660 \@@_define_opentype_onoffreset:nnnnn {Vertical} {Alternates}
                                                                             {vkna} {vkna} {+hkna}
2661 \@@_define_opentype_onoffreset:nnnnn {Vertical} {KanaAlternates}
                                                                             {vkrn} {vkrn} {}
2662 \@@_define_opentype_onoffreset:nnnnn {Vertical} {Kerning}
                                                                             {valt} {valt} {+vhal,
2663 \@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternateMetrics}
2664 \@@_define_opentype_onoffreset:nnnnn {Vertical} {HalfMetrics}
                                                                             {vhal} {vhal} {+valt,
2665 \@@_define_opentype_onoffreset:nnnnn {Vertical} {ProportionalMetrics}
                                                                             {vpal} {vpal} {+valt,
```

OpenType features that need numbering

38.3.1 Alternate

```
2666 \@@_define_opentype_feature_group:n {Alternate}
2667 \keys_define:nn {fontspec-opentype}
    Alternate .default:n = {o},
2670 (luatex) Alternate / Random .code:n =
           { \@@ make OT feature:nnn {salt}{ +salt = random }{} } ,
```

```
Alternate / unknown .code:n =
2672
2673
2674
       \clist_map_inline:nn {#1}
         2676
2677 }
2678 \aliasfontfeature{Alternate}{StylisticAlternates}
 38.3.2 Variant / StylisticSet
2679 \@@_define_opentype_feature_group:n {Variant}
2680 \keys_define:nn {fontspec-opentype}
2681 {
     Variant .default:n = \{o\} ,
2682
     Variant / unknown .code:n =
2683
2684
       \clist_map_inline:nn {#1}
2685
2686
         {
           \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
2687
2688
      }
2689
2690 }
2691 \aliasfontfeature{Variant}{StylisticSet}
 38.3.3 CharacterVariant
2692 \@@_define_opentype_feature_group:n {CharacterVariant}
2693 \use:x
2694 {
     \cs_new:Npn \exp_not:N \fontspec_parse_cv:w
2695
         ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
2696
2697
        \@@_make_OT_feature:xxx
2698
          { cv \exp_not:N \two@digits {##1} } { +cv \exp_not:N \two@digits {##1} = ##2 } {}
2699
2700
     \keys define:nn {fontspec-opentype}
2701
      {
2702
       CharacterVariant / unknown .code:n =
2703
        {
2704
         \clist_map_inline:nn {##1}
2705
2706
           \exp_not:N \fontspec_parse_cv:w
             ####1 \c_colon_str o \c_colon_str \exp_not:N \q_nil
2709
2710
2711
2712 }
 Possibilities: a:o:\q_nil or a:b:o:\q_nil.
 38.3.4 Annotation
2713 \@@_define_opentype_feature_group:n {Annotation}
2714 \keys define:nn {fontspec-opentype}
2715 {
```

```
Annotation .default:n = {o} ,
2716
      Annotation / unknown .code:n =
2717
2718
        \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
2720
2721 }
 38.3.5 Ornament
2722 \@@_define_opentype_feature_group:n {Ornament}
2723 \keys_define:nn {fontspec-opentype}
2724 {
     Ornament .default:n = {o} ,
2725
     Ornament / unknown .code:n =
        \@@_make_OT_feature:nnn {ornm} { +ornm=#1 } {}
2728
2729
2730 }
         Script and Language
 38.4
 38.4.1 Script
2731 \keys_define:nn { fontspec-opentype } { Script .choice: }
2732 \cs_new:Nn \fontspec_new_script:nn
2733 {
     \keys_define:nn { fontspec-opentype } { Script / #1 .code:n =
2734
        \bool_set_false:N \l_@@_script_exist_bool
2735
        \clist_map_inline:nn {#2}
2736
         {
2737
          \@@_check_script:nTF {####1}
2738
2739
            \tl set:Nn \l fontspec script tl {####1}
2740
            \int set:Nn \l @@ script int {\l @@ strnum int}
2741
            \bool_set_true:N \l_@@_script_exist_bool
2742
            \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
2743
            \clist_map_break:
2744
           }
2745
           { }
2746
         }
2747
        \bool_if:NF \l_@@_script_exist_bool
2748
2749
          \str_if_eq:nnTF {#1} {Latin}
2750
2751
            \@@_warning:nx {script-not-exist} {#1}
2752
           }
2753
           {
2754
            \@@_check_script:nTF {latn}
2755
2756
              \@@_warning:nx {script-not-exist-latn} {#1}
2757
              \tl set:Nn \l fontspec script tl {latn}
2758
              \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
2759
2760
```

2761

{

```
\@@_warning:nx {script-not-exist} {#1}
2762
2763
2764
2765
2766
      }
2767 }
 38.4.2 Language
2768 keys_define:nn { fontspec-opentype } { Language .choice: }
2769 \cs_new:Nn \fontspec_new_lang:nn
2770 {
     \keys define:nn { fontspec-opentype } { Language / #1 .code:n =
2771
     \@@_check_lang:nTF {#2}
2772
2773
          \tl_set:Nn \l_fontspec_lang_tl {#2}
2774
          \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
2775
          \tl_gset:Nx \g_00_single_feat_tl { language=#2 }
2776
2777
         {
2778
2779
          \@@_warning:nx {language-not-exist} {#1}
2780
         \keys_set:nn { fontspec-opentype } { Language = Default }
2781
2782
2783 }
 Default
2784 \@@_keys_define_code:nnn {fontspec-opentype}{ Language / Default }
2785 {
     \tl_set:Nn \l_fontspec_lang_tl {DFLT}
     \int_zero:N \l_@@_language_int
     \tl_gset:Nn \g_@0_single_feat_tl { language=DFLT }
2789 }
```

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

```
2790 \keys_define:nn {fontspec-opentype}
2791 {
     Language / Turkish .code:n =
2792
2793
        \@@_check_lang:nTF {TRK}
2794
2795
          \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
2796
          \tl_set:Nn \l_fontspec_lang_tl {TRK}
2797
          \tl_gset:Nn \g_@0_single_feat_tl { language=TRK }
2798
         }
2799
         {
2800
          \@@_check_lang:nTF {TUR}
2801
2802
2803
            \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
2804
            \tl_set:Nn \l_fontspec_lang_tl {TUR}
2805
            \tl_gset:Nn \g_@0_single_feat_tl { language=TUR }
```

38.5 Backwards compatibility

Backwards compatibility:

```
2814 \cs_new:Nn \@@_ot_compat:nn
2815
     {
2816
        \aliasfontfeatureoption {#1} {#20ff} {No#2}
     }
2818 \@@_ot_compat:nn {Ligatures}
                                    {Rare}
2819 \@@_ot_compat:nn {Ligatures}
                                    {Required}
2820 \@@ ot compat:nn {Ligatures}
                                    {Common}
                                    {Discretionary}
2821 \00 ot compat:nn {Ligatures}
2822 \@@_ot_compat:nn {Ligatures}
                                    {Contextual}
2823 \@@_ot_compat:nn {Ligatures}
                                    {Historic}
                                    {SlashedZero}
2824 \@@_ot_compat:nn {Numbers}
2825 \@@_ot_compat:nn {Contextuals} {Swash}
2826 \@@_ot_compat:nn {Contextuals} {Alternate}
2827 \@@_ot_compat:nn {Contextuals} {WordInitial}
2828 \@@ ot compat:nn {Contextuals} {WordFinal}
2829 \@@_ot_compat:nn {Contextuals} {LineFinal}
2830 \@@_ot_compat:nn {Contextuals} {Inner}
2831 \@@_ot_compat:nn {Diacritics} {MarkToBase}
2832 \@@_ot_compat:nn {Diacritics}
                                   {MarkToMark}
2833 \@@_ot_compat:nn {Diacritics} {AboveBase}
2834 \@@_ot_compat:nn {Diacritics} {BelowBase}
```

38.6 Font script definitions

```
2835 \newfontscript{Arabic}{arab}
2836 \newfontscript{Armenian}{armn}
2837 \newfontscript{Balinese}{bali}
2838 \newfontscript{Bengali}{bng2,beng}
2839 \newfontscript{Bengali}{bng2,beng}
2840 \newfontscript{Bopomofo}{bopo}
2840 \newfontscript{Braille}{brai}
2841 \newfontscript{Buginese}{bugi}
2842 \newfontscript{Buhid}{buhd}
2843 \newfontscript{Byzantine~Music}{byzm}
2844 \newfontscript{Canadian~Syllabics}{cans}
2845 \newfontscript{Cherokee}{cher}
2846 \newfontscript{CJK~Ideographic}{hani}
2847 \newfontscript{Coptic}{copt}
2848 \newfontscript{Cypriot~Syllabary}{cprt}
2849 \newfontscript{Cyrillic}{cyrl}
```

```
2850 \newfontscript{Default}{DFLT}
2851 \newfontscript{Deseret}{dsrt}
2852 \newfontscript{Devanagari}{dev2,deva}
2853 \newfontscript{Ethiopic}{ethi}
2854 \newfontscript{Georgian}{geor}
2855 \newfontscript{Glagolitic}{glag}
2856 \newfontscript{Gothic}{goth}
2857 \newfontscript{Greek}{grek}
2858 \newfontscript{Gujarati}{gjr2,gujr}
2859 \newfontscript{Gurmukhi}{gur2,guru}
2860 \newfontscript{Hangul~Jamo}{jamo}
2861 \newfontscript{Hangul}{hang}
2862 \newfontscript{Hanunoo}{hano}
2863 \newfontscript{Hebrew}{hebr}
2864 \newfontscript{Hiragana~and~Katakana}{kana}
2865 \newfontscript{Javanese}{java}
2866 \newfontscript{Kannada}{knd2,knda}
2867 \newfontscript{Kharosthi}{khar}
2868 \newfontscript{Khmer}{khmr}
{\tt 2869 \backslash newfontscript\{Lao\}\{lao~\}}
2871 \newfontscript{Limbu}{limb}
2872 \newfontscript{Linear~B}{linb}
2873 \newfontscript{Malayalam}{mlm2,mlym}
2874 \newfontscript{Math}{math}
2875 \newfontscript{Mongolian}{mong}
2876 \newfontscript{Musical~Symbols}{musc}
2877 \newfontscript{Myanmar}{mymr}
2878 \newfontscript{N'ko}{nko~}
2879 \newfontscript{Ogham}{ogam}
2880 \newfontscript{Old~Italic}{ital}
2881 \newfontscript{Old~Persian~Cuneiform}{xpeo}
2882 \newfontscript{Oriya}{ory2,orya}
2883 \newfontscript{Osmanya}{osma}
2884 \newfontscript{Phags-pa}{phag}
2885 \newfontscript{Phoenician}{phnx}
2886 \newfontscript{Runic}{runr}
2887 \newfontscript{Shavian}{shaw}
2888 \newfontscript{Sinhala}{sinh}
2889 \newfontscript{Sumero-Akkadian~Cuneiform}{xsux}
2890 \newfontscript{Syloti~Nagri}{sylo}
2891 \newfontscript{Syriac}{syrc}
2892 \newfontscript{Tagalog}{tglg}
2893 \newfontscript{Tagbanwa}{tagb}
2894 \newfontscript{Tai~Le}{tale}
2895 \newfontscript{Tai~Lu}{talu}
2896 \newfontscript{Tamil}{tml2,taml}
2897 \newfontscript{Telugu}{tel2,telu}
2898 \newfontscript{Thaana}{thaa}
2899 \newfontscript{Thai}{thai}
2900 \newfontscript{Tibetan}{tibt}
```

```
2901 \newfontscript{Tifinagh}{tfng}
2902 \newfontscript{Ugaritic~Cuneiform}{ugar}
2903 \newfontscript{Yi}{yi~~}
```

For convenience:

2904 \newfontscript{Kana}{kana}
2905 \newfontscript{Maths}{math}
2906 \newfontscript{CJK}{hani}

38.7 Font language definitions

```
2907 \newfontlanguage{Abaza}{ABA}
2908 \newfontlanguage{Abkhazian}{ABK}
2909 \newfontlanguage{Adyghe}{ADY}
2910 \newfontlanguage{Afrikaans}{AFK}
2911 \newfontlanguage{Afar}{AFR}
2912 \newfontlanguage{Agaw}{AGW}
2913 \newfontlanguage{Altai}{ALT}
2914 \newfontlanguage{Amharic}{AMH}
2915 \newfontlanguage{Arabic}{ARA}
2916 \newfontlanguage{Aari}{ARI}
2917 \newfontlanguage{Arakanese}{ARK}
2918 \newfontlanguage{Assamese}{ASM}
2919 \newfontlanguage{Athapaskan}{ATH}
2920 \newfontlanguage{Avar}{AVR}
2921 \newfontlanguage{Awadhi}{AWA}
2922 \newfontlanguage{Aymara}{AYM}
2923 \newfontlanguage{Azeri}{AZE}
2924 \newfontlanguage{Badaga}{BAD}
2925 \newfontlanguage{Baghelkhandi}{BAG}
2926 \newfontlanguage{Balkar}{BAL}
2927 \newfontlanguage{Baule}{BAU}
2928 \newfontlanguage{Berber}{BBR}
2929 \newfontlanguage{Bench}{BCH}
2930 \newfontlanguage{Bible~Cree}{BCR}
2931 \newfontlanguage{Belarussian}{BEL}
2932 \newfontlanguage{Bemba}{BEM}
2933 \newfontlanguage{Bengali}{BEN}
2934 \newfontlanguage{Bulgarian}{BGR}
2935 \newfontlanguage{Bhili}{BHI}
2936 \newfontlanguage{Bhojpuri}{BHO}
2937 \newfontlanguage{Bikol}{BIK}
2938 \newfontlanguage{Bilen}{BIL}
2939 \newfontlanguage{Blackfoot}{BKF}
2940 \newfontlanguage{Balochi}{BLI}
2941 \newfontlanguage{Balante}{BLN}
2942 \newfontlanguage{Balti}{BLT}
2943 \newfontlanguage{Bambara}{BMB}
2944 \newfontlanguage{Bamileke}{BML}
2945 \newfontlanguage{Breton}{BRE}
2946 \newfontlanguage{Brahui}{BRH}
2947 \newfontlanguage{Braj~Bhasha}{BRI}
2948 \newfontlanguage{Burmese}{BRM}
```

```
2949 \newfontlanguage{Bashkir}{BSH}
2950 \newfontlanguage{Beti}{BTI}
2951 \newfontlanguage{Catalan}{CAT}
2952 \newfontlanguage{Cebuano}{CEB}
2953 \newfontlanguage{Chechen}{CHE}
2954 \newfontlanguage{Chaha~Gurage}{CHG}
2955 \newfontlanguage{Chattisgarhi}{CHH}
2956 \newfontlanguage{Chichewa}{CHI}
2957 \newfontlanguage{Chukchi}{CHK}
2958 \newfontlanguage{Chipewyan}{CHP}
2959 \newfontlanguage{Cherokee}{CHR}
2960 \newfontlanguage{Chuvash}{CHU}
2961 \newfontlanguage{Comorian}{CMR}
2962 \newfontlanguage{Coptic}{COP}
2963 \newfontlanguage{Cree}{CRE}
2964 \newfontlanguage{Carrier}{CRR}
2965 \newfontlanguage{Crimean~Tatar}{CRT}
2966 \newfontlanguage{Church~Slavonic}{CSL}
2967 \newfontlanguage{Czech}{CSY}
2968 \newfontlanguage{Danish}{DAN}
2969 \newfontlanguage{Dargwa}{DAR}
2970 \newfontlanguage{Woods~Cree}{DCR}
2971 \newfontlanguage{German}{DEU}
2972 \newfontlanguage{Dogri}{DGR}
2973 \newfontlanguage{Divehi}{DIV}
2974 \newfontlanguage{Djerma}{DJR}
2975 \newfontlanguage{Dangme}{DNG}
2976 \newfontlanguage{Dinka}{DNK}
2977 \newfontlanguage{Dungan}{DUN}
2978 \newfontlanguage{Dzongkha}{DZN}
2979 \newfontlanguage{Ebira}{EBI}
2980 \newfontlanguage{Eastern~Cree}{ECR}
2981 \newfontlanguage{Edo}{EDO}
2982 \newfontlanguage{Efik}{EFI}
2983 \newfontlanguage{Greek}{ELL}
2984 \newfontlanguage{English}{ENG}
2985 \newfontlanguage{Erzya}{ERZ}
2986 \newfontlanguage{Spanish}{ESP}
2987 \newfontlanguage{Estonian}{ETI}
2988 \newfontlanguage{Basque}{EUQ}
2989 \newfontlanguage{Evenki}{EVK}
2990 \newfontlanguage{Even}{EVN}
2991 \newfontlanguage{Ewe}{EWE}
2992 \newfontlanguage{French~Antillean}{FAN}
2993 \newfontlanguage{Farsi}{FAR}
2994 \newfontlanguage{Parsi}{FAR}
2995 \newfontlanguage{Persian}{FAR}
2996 \newfontlanguage{Finnish}{FIN}
2997 \newfontlanguage{Fijian}{FJI}
2998 \newfontlanguage{Flemish}{FLE}
2999 \newfontlanguage{Forest~Nenets}{FNE}
```

```
3000 \newfontlanguage{Fon}{FON}
3001 \newfontlanguage{Faroese}{FOS}
3002 \newfontlanguage{French}{FRA}
3003 \newfontlanguage{Frisian}{FRI}
3004 \newfontlanguage{Friulian}{FRL}
3005 \newfontlanguage{Futa}{FTA}
3006 \newfontlanguage{Fulani}{FUL}
3007 \newfontlanguage{Ga}{GAD}
3008 \newfontlanguage{Gaelic}{GAE}
3009 \newfontlanguage{Gagauz}{GAG}
3010 \newfontlanguage{Galician}{GAL}
3011 \newfontlanguage{Garshuni}{GAR}
3012 \newfontlanguage{Garhwali}{GAW}
3013 \newfontlanguage{Ge'ez}{GEZ}
3014 \newfontlanguage{Gilyak}{GIL}
3015 \newfontlanguage{Gumuz}{GMZ}
3016 \newfontlanguage{Gondi}{GON}
3017 \newfontlanguage{Greenlandic}{GRN}
3018 \newfontlanguage{Garo}{GRO}
3019 \newfontlanguage{Guarani}{GUA}
3020 \newfontlanguage{Gujarati}{GUJ}
3021 \newfontlanguage{Haitian}{HAI}
3022 \newfontlanguage{Halam}{HAL}
3023 \newfontlanguage{Harauti}{HAR}
3024 \newfontlanguage{Hausa}{HAU}
3025 \newfontlanguage{Hawaiin}{HAW}
3026 \newfontlanguage{Hammer-Banna}{HBN}
3027 \newfontlanguage{Hiligaynon}{HIL}
3028 \newfontlanguage{Hindi}{HIN}
3029 \newfontlanguage{High~Mari}{HMA}
3030 \newfontlanguage{Hindko}{HND}
3031 \newfontlanguage{Ho}{HO}
3032 \newfontlanguage{Harari}{HRI}
3033 \newfontlanguage{Croatian}{HRV}
3034 \newfontlanguage{Hungarian}{HUN}
3035 \newfontlanguage{Armenian}{HYE}
3036 \newfontlanguage{Igbo}{IBO}
3037 \newfontlanguage{Ijo}{IJO}
3038 \newfontlanguage{Ilokano}{ILO}
3039 \newfontlanguage{Indonesian}{IND}
3040 \newfontlanguage{Ingush}{ING}
3041 \newfontlanguage{Inuktitut}{INU}
3042 \newfontlanguage{Irish}{IRI}
3043 \newfontlanguage{Irish~Traditional}{IRT}
3044 \newfontlanguage{Icelandic}{ISL}
3045 \newfontlanguage{Inari~Sami}{ISM}
3046 \newfontlanguage{Italian}{ITA}
3047 \newfontlanguage{Hebrew}{IWR}
3048 \newfontlanguage{Javanese}{JAV}
3049 \newfontlanguage{Yiddish}{JII}
3050 \newfontlanguage{Japanese}{JAN}
```

```
3051 \newfontlanguage{Judezmo}{JUD}
3052 \newfontlanguage{Jula}{JUL}
3053 \newfontlanguage{Kabardian}{KAB}
3054 \newfontlanguage{Kachchi}{KAC}
3055 \newfontlanguage{Kalenjin}{KAL}
3056 \newfontlanguage{Kannada}{KAN}
3057 \newfontlanguage{Karachay}{KAR}
3058 \newfontlanguage{Georgian}{KAT}
3059 \newfontlanguage{Kazakh}{KAZ}
3060 \newfontlanguage{Kebena}{KEB}
3061 \newfontlanguage{Khutsuri~Georgian}{KGE}
3062 \newfontlanguage{Khakass}{KHA}
3063 \newfontlanguage{Khanty-Kazim}{KHK}
3064 \newfontlanguage{Khmer}{KHM}
3065 \newfontlanguage{Khanty-Shurishkar}{KHS}
3066 \newfontlanguage{Khanty-Vakhi}{KHV}
3067 \newfontlanguage{Khowar}{KHW}
3068 \newfontlanguage{Kikuyu}{KIK}
3069 \newfontlanguage{Kirghiz}{KIR}
3070 \newfontlanguage{Kisii}{KIS}
3071 \newfontlanguage{Kokni}{KKN}
3072 \newfontlanguage{Kalmyk}{KLM}
3073 \newfontlanguage{Kamba}{KMB}
3074 \newfontlanguage{Kumaoni}{KMN}
3075 \newfontlanguage{Komo}{KMO}
3076 \newfontlanguage{Komso}{KMS}
3077 \newfontlanguage{Kanuri}{KNR}
3078 \newfontlanguage{Kodagu}{KOD}
3079 \newfontlanguage{Korean~Old~Hangul}{KOH}
3080 \newfontlanguage{Konkani}{KOK}
3081 \newfontlanguage{Kikongo}{KON}
3082 \newfontlanguage{Komi-Permyak}{KOP}
3083 \newfontlanguage{Korean}{KOR}
3084 \neq Minus (KOZ)
3085 \newfontlanguage{Kpelle}{KPL}
3086 \newfontlanguage{Krio}{KRI}
3087 \newfontlanguage{Karakalpak}{KRK}
3088 \newfontlanguage{Karelian}{KRL}
3089 \newfontlanguage{Karaim}{KRM}
3090 \newfontlanguage{Karen}{KRN}
3091 \newfontlanguage{Koorete}{KRT}
3092 \newfontlanguage{Kashmiri}{KSH}
3093 \newfontlanguage{Khasi}{KSI}
3094 \newfontlanguage{Kildin~Sami}{KSM}
3095 \newfontlanguage{Kui}{KUI}
3096 \newfontlanguage{Kulvi}{KUL}
3097 \newfontlanguage{Kumyk}{KUM}
3098 \newfontlanguage{Kurdish}{KUR}
3099 \newfontlanguage{Kurukh}{KUU}
3100 \newfontlanguage{Kuy}{KUY}
```

3101 \newfontlanguage{Koryak}{KYK}

```
3102 \newfontlanguage{Ladin}{LAD}
3103 \newfontlanguage{Lahuli}{LAH}
3104 \newfontlanguage{Lak}{LAK}
3105 \newfontlanguage{Lambani}{LAM}
3106 \newfontlanguage{Lao}{LAO}
3107 \newfontlanguage{Latin}{LAT}
3108 \newfontlanguage{Laz}{LAZ}
3109 \newfontlanguage{L-Cree}{LCR}
3110 \newfontlanguage{Ladakhi}{LDK}
3111 \newfontlanguage{Lezgi}{LEZ}
3112 \newfontlanguage{Lingala}{LIN}
3113 \newfontlanguage{Low~Mari}{LMA}
3114 \newfontlanguage{Limbu}{LMB}
3115 \newfontlanguage{Lomwe}{LMW}
3116 \newfontlanguage{Lower~Sorbian}{LSB}
3117 \newfontlanguage{Lule~Sami}{LSM}
3118 \newfontlanguage{Lithuanian}{LTH}
3119 \newfontlanguage{Luba}{LUB}
3120 \newfontlanguage{Luganda}{LUG}
3121 \newfontlanguage{Luhya}{LUH}
3122 \newfontlanguage{Luo}{LUO}
3123 \newfontlanguage{Latvian}{LVI}
3124 \newfontlanguage{Majang}{MAJ}
3125 \newfontlanguage{Makua}{MAK}
3126 \newfontlanguage{Malayalam~Traditional}{MAL}
3127 \newfontlanguage{Mansi}{MAN}
3128 \newfontlanguage{Marathi}{MAR}
3129 \newfontlanguage{Marwari}{MAW}
3130 \newfontlanguage{Mbundu}{MBN}
3131 \newfontlanguage{Manchu}{MCH}
3132 \newfontlanguage{Moose~Cree}{MCR}
3133 \newfontlanguage{Mende}{MDE}
3134 \newfontlanguage{Me'en}{MEN}
3135 \newfontlanguage{Mizo}{MIZ}
3136 \newfontlanguage{Macedonian}{MKD}
3137 \newfontlanguage{Male}{MLE}
3138 \newfontlanguage{Malagasy}{MLG}
3139 \newfontlanguage{Malinke}{MLN}
3140 \newfontlanguage{Malayalam~Reformed}{MLR}
3141 \newfontlanguage{Malay}{MLY}
3142 \newfontlanguage{Mandinka}{MND}
3143 \newfontlanguage{Mongolian}{MNG}
3144 \newfontlanguage{Manipuri}{MNI}
3145 \newfontlanguage{Maninka}{MNK}
3146 \newfontlanguage{Manx~Gaelic}{MNX}
3147 \newfontlanguage{Moksha}{MOK}
3148 \newfontlanguage{Moldavian}{MOL}
3149 \newfontlanguage{Mon}{MON}
3150 \newfontlanguage{Moroccan}{MOR}
3151 \newfontlanguage{Maori}{MRI}
3152 \newfontlanguage{Maithili}{MTH}
```

```
3153 \newfontlanguage{Maltese}{MTS}
3154 \newfontlanguage{Mundari}{MUN}
3155 \newfontlanguage{Naga-Assamese}{NAG}
3156 \newfontlanguage{Nanai}{NAN}
3157 \newfontlanguage{Naskapi}{NAS}
3158 \newfontlanguage{N-Cree}{NCR}
3159 \newfontlanguage{Ndebele}{NDB}
3160 \newfontlanguage{Ndonga}{NDG}
3161 \newfontlanguage{Nepali}{NEP}
3162 \newfontlanguage{Newari}{NEW}
3163 \newfontlanguage{Nagari}{NGR}
3164 \newfontlanguage{Norway~House~Cree}{NHC}
3165 \newfontlanguage{Nisi}{NIS}
3166 \newfontlanguage{Niuean}{NIU}
3167 \newfontlanguage{Nkole}{NKL}
3168 \newfontlanguage{N'ko}{NKO}
3169 \newfontlanguage{Dutch}{NLD}
3170 \newfontlanguage{Nogai}{NOG}
3171 \newfontlanguage{Norwegian}{NOR}
3172 \newfontlanguage{Northern~Sami}{NSM}
3173 \newfontlanguage{Northern~Tai}{NTA}
3174 \newfontlanguage{Esperanto}{NTO}
3175 \newfontlanguage{Nynorsk}{NYN}
3176 \newfontlanguage{Oji-Cree}{OCR}
3177 \newfontlanguage{Ojibway}{OJB}
3178 \newfontlanguage{Oriya}{ORI}
3179 \newfontlanguage{Oromo}{ORO}
3180 \newfontlanguage{Ossetian}{OSS}
3181 \newfontlanguage{Palestinian~Aramaic}{PAA}
3182 \newfontlanguage{Pali}{PAL}
3183 \newfontlanguage{Punjabi}{PAN}
3184 \newfontlanguage{Palpa}{PAP}
3185 \newfontlanguage{Pashto}{PAS}
3186 \mbox{ } \mbox{newfontlanguage{Polytonic~Greek}{PGR}
3187 \newfontlanguage{Pilipino}{PIL}
3188 \newfontlanguage{Palaung}{PLG}
3189 \newfontlanguage{Polish}{PLK}
3190 \newfontlanguage{Provencal}{PRO}
3191 \newfontlanguage{Portuguese}{PTG}
3192 \newfontlanguage{Chin}{QIN}
3193 \newfontlanguage{Rajasthani}{RAJ}
3194 \newfontlanguage{R-Cree}{RCR}
3195 \newfontlanguage{Russian~Buriat}{RBU}
3196 \newfontlanguage{Riang}{RIA}
3197 \newfontlanguage{Rhaeto-Romanic}{RMS}
3198 \newfontlanguage{Romanian}{ROM}
3199 \newfontlanguage{Romany}{ROY}
3200 \newfontlanguage{Rusyn}{RSY}
3201 \newfontlanguage{Ruanda}{RUA}
3202 \newfontlanguage{Russian}{RUS}
```

3203 \newfontlanguage{Sadri}{SAD}

```
3204 \newfontlanguage{Sanskrit}{SAN}
3205 \newfontlanguage{Santali}{SAT}
3206 \newfontlanguage{Sayisi}{SAY}
3207 \newfontlanguage{Sekota}{SEK}
3208 \newfontlanguage{Selkup}{SEL}
3209 \newfontlanguage{Sango}{SGO}
3210 \newfontlanguage{Shan}{SHN}
3211 \newfontlanguage{Sibe}{SIB}
3212 \newfontlanguage{Sidamo}{SID}
3213 \newfontlanguage{Silte~Gurage}{SIG}
3214 \newfontlanguage{Skolt~Sami}{SKS}
3215 \newfontlanguage{Slovak}{SKY}
3216 \newfontlanguage{Slavey}{SLA}
3217 \newfontlanguage{Slovenian}{SLV}
3218 \newfontlanguage{Somali}{SML}
3219 \newfontlanguage{Samoan}{SMO}
3220 \newfontlanguage{Sena}{SNA}
3221 \newfontlanguage{Sindhi}{SND}
3222 \newfontlanguage{Sinhalese}{SNH}
3223 \newfontlanguage{Soninke}{SNK}
3224 \newfontlanguage{Sodo~Gurage}{SOG}
3225 \newfontlanguage{Sotho}{SOT}
3226 \newfontlanguage{Albanian}{SQI}
3227 \newfontlanguage{Serbian}{SRB}
3228 \newfontlanguage{Saraiki}{SRK}
3229 \newfontlanguage{Serer}{SRR}
3230 \newfontlanguage{South~Slavey}{SSL}
3231 \newfontlanguage{Southern~Sami}{SSM}
3232 \newfontlanguage{Suri}{SUR}
3233 \newfontlanguage{Svan}{SVA}
3234 \newfontlanguage{Swedish}{SVE}
{\tt 3235} \verb| newfontlanguage{Swadaya~Aramaic}{SWA}|
3236 \newfontlanguage{Swahili}{SWK}
3237 \newfontlanguage{Swazi}{SWZ}
3238 \newfontlanguage{Sutu}{SXT}
3239 \newfontlanguage{Syriac}{SYR}
3240 \newfontlanguage{Tabasaran}{TAB}
3241 \newfontlanguage{Tajiki}{TAJ}
3242 \newfontlanguage{Tamil}{TAM}
3243 \newfontlanguage{Tatar}{TAT}
3244 \newfontlanguage{TH-Cree}{TCR}
3245 \newfontlanguage{Telugu}{TEL}
3246 \newfontlanguage{Tongan}{TGN}
3247 \newfontlanguage{Tigre}{TGR}
3248 \newfontlanguage{Tigrinya}{TGY}
3249 \newfontlanguage{Thai}{THA}
3250 \newfontlanguage{Tahitian}{THT}
3251 \newfontlanguage{Tibetan}{TIB}
3252 \newfontlanguage{Turkmen}{TKM}
3253 \newfontlanguage{Temne}{TMN}
3254 \newfontlanguage{Tswana}{TNA}
```

```
3255 \newfontlanguage{Tundra~Nenets}{TNE}
3256 \newfontlanguage{Tonga}{TNG}
3257 \newfontlanguage{Todo}{TOD}
3258 \newfontlanguage{Tsonga}{TSG}
3259 \newfontlanguage{Turoyo~Aramaic}{TUA}
3260 \newfontlanguage{Tulu}{TUL}
3261 \newfontlanguage{Tuvin}{TUV}
3262 \newfontlanguage{Twi}{TWI}
3263 \newfontlanguage{Udmurt}{UDM}
3264 \newfontlanguage{Ukrainian}{UKR}
3265 \newfontlanguage{Urdu}{URD}
3266 \newfontlanguage{Upper~Sorbian}{USB}
3267 \newfontlanguage{Uyghur}{UYG}
3268 \newfontlanguage{Uzbek}{UZB}
3269 \newfontlanguage{Venda}{VEN}
3270 \newfontlanguage{Vietnamese}{VIT}
3271 \newfontlanguage{Wa}{WA}
3272 \newfontlanguage{Wagdi}{WAG}
3273 \newfontlanguage{West-Cree}{WCR}
3274 \newfontlanguage{Welsh}{WEL}
3275 \newfontlanguage{Wolof}{WLF}
3276 \newfontlanguage{Tai~Lue}{XBD}
3277 \newfontlanguage{Xhosa}{XHS}
3278 \newfontlanguage{Yakut}{YAK}
3279 \newfontlanguage{Yoruba}{YBA}
3280 \newfontlanguage{Y-Cree}{YCR}
3281 \newfontlanguage{Yi~Classic}{YIC}
3282 \newfontlanguage{Yi~Modern}{YIM}
3283 \newfontlanguage{Chinese~Hong~Kong}{ZHH}
3284 \newfontlanguage{Chinese~Phonetic}{ZHP}
3285 \newfontlanguage{Chinese~Simplified}{ZHS}
3286 \newfontlanguage{Chinese~Traditional}{ZHT}
3287 \newfontlanguage{Zande}{ZND}
3288 \newfontlanguage{Zulu}{ZUL}
```

38.8 AAT feature definitions

These are only defined for $X_{\overline{1}}T_{\overline{1}}X$.

38.8.1 Ligatures

```
3289 \@@_define_aat_feature_group:n {Ligatures}
3290 \@@_define_aat_feature:nnnn
                                      {Ligatures} {Required} {1} {0}
3291 \00 define aat feature:nnnn
                                      {Ligatures} {NoRequired} {1} {1}
3292 \00 define aat feature:nnnn
                                      {Ligatures} {Common} {1} {2}
                                      {Ligatures} {NoCommon} {1} {3}
3293 \@@ define aat feature:nnnn
3294 \00 define aat feature:nnnn
                                      {Ligatures} {Rare} {1} {4}
3295 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoRare} {1} {5}
                                      {Ligatures} {Discretionary} {1} {4}
3296 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoDiscretionary} {1} {5}
3297 \@@_define_aat_feature:nnnn
                                      {Ligatures} {Logos} {1} {6}
3298 \@@_define_aat_feature:nnnn
3299 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoLogos} {1} {7}
```

```
{Ligatures} {Rebus} {1} {8}
3300 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoRebus} {1} {9}
3301 \@@_define_aat_feature:nnnn
3302 \@@_define_aat_feature:nnnn
                                      {Ligatures} {Diphthong} {1} {10}
3303 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoDiphthong} {1} {11}
3304 \@@_define_aat_feature:nnnn
                                      {Ligatures} {Squared} {1} {12}
3305 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoSquared} {1} {13}
3306 \@@_define_aat_feature:nnnn
                                      {Ligatures} {AbbrevSquared} {1} {14}
3307 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoAbbrevSquared} {1} {15}
3308 \@@_define_aat_feature:nnnn
                                      {Ligatures} {Icelandic} {1} {32}
3309 \@@_define_aat_feature:nnnn
                                      {Ligatures} {NoIcelandic} {1} {33}
 Emulate CM extra ligatures.
3310 keys_define:nn {fontspec-aat}
    Ligatures / TeX .code:n =
3312
3313
       \@@_update_featstr:n { mapping = tex-text }
3314
3315
3316 }
 38.8.2 Letters
3317 \00_define_aat_feature_group:n {Letters}
3318 \@@_define_aat_feature:nnnn
                                      {Letters} {Normal} {3} {0}
                                      {Letters} {Uppercase} {3} {1}
3319 \@@_define_aat_feature:nnnn
                                      {Letters} {Lowercase} {3} {2}
3320 \@@_define_aat_feature:nnnn
                                      {Letters} {SmallCaps} {3} {3}
3321 \@@_define_aat_feature:nnnn
3322 \@@_define_aat_feature:nnnn
                                      {Letters} {InitialCaps} {3} {4}
```

38.8.3 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.

```
3323 \@@_define_aat_feature_group:n {Numbers}
3324 \@@_define_aat_feature:nnnn
                                      {Numbers} {Monospaced} {6} {0}
3325 \@@_define_aat_feature:nnnn
                                      {Numbers} {Proportional} {6} {1}
3326 \@@ define aat feature:nnnn
                                      {Numbers} {Lowercase} {21} {0}
3327 \@@_define_aat_feature:nnnn
                                      {Numbers} {OldStyle} {21} {0}
3328 \@@_define_aat_feature:nnnn
                                      {Numbers} {Uppercase} {21} {1}
3329 \@@_define_aat_feature:nnnn
                                      {Numbers} {Lining} {21} {1}
3330 \@@_define_aat_feature:nnnn
                                      {Numbers} {SlashedZero} {14} {5}
3331 \@@_define_aat_feature:nnnn
                                      {Numbers} {NoSlashedZero} {14} {4}
 38.8.4 Contextuals
```

```
{Contextuals} {LineFinal} {8} {6}
3339 \@@_define_aat_feature:nnnn
                                      {Contextuals} {NoLineFinal} {8} {7}
3340 \@@_define_aat_feature:nnnn
                                      {Contextuals} {Inner} {8} {8}
3341 \@@_define_aat_feature:nnnn
                                      {Contextuals} {NoInner} {8} {9}
_{334^2}\@0_define_aat_feature:nnnn
 38.8.5 Diacritics
3343 \@@_define_aat_feature_group:n {Diacritics}
3344 \@@_define_aat_feature:nnnn
                                      {Diacritics} {Show} {9} {0}
3345 \@@_define_aat_feature:nnnn
                                      {Diacritics} {Hide} {9} {1}
                                      {Diacritics} {Decompose} {9} {2}
3346 \@@_define_aat_feature:nnnn
 38.8.6 Vertical position
3347 \@@_define_aat_feature_group:n {VerticalPosition}
                                      {VerticalPosition} {Normal} {10} {0}
3348 \@@_define_aat_feature:nnnn
                                      {VerticalPosition} {Superior} {10} {1}
3349 \@@_define_aat_feature:nnnn
                                      {VerticalPosition} {Inferior} {10} {2}
3350 \@@_define_aat_feature:nnnn
3351 \@@_define_aat_feature:nnnn
                                      {VerticalPosition} {Ordinal} {10} {3}
 38.8.7 Fractions
3352 \@@_define_aat_feature_group:n {Fractions}
3353 \@@_define_aat_feature:nnnn
                                      {Fractions} {On} {11} {1}
3354 \@@_define_aat_feature:nnnn
                                      {Fractions} {Off} {11} {0}
3355 \@@_define_aat_feature:nnnn
                                      {Fractions} {Diagonal} {11} {2}
 38.8.8 Alternate
3356 \@@_define_aat_feature_group:n { Alternate }
3357 \keys_define:nn {fontspec-aat}
3358 {
3359
     Alternate .default:n = {o} ,
3360
     Alternate / unknown .code:n =
3361
        \clist_map_inline:nn {#1}
3362
          {
3363
            \@@_make_AAT_feature:nn {17}{##1}
3364
3365
3366
       }
3367 }
 38.8.9 Variant / StylisticSet
3368 \@@_define_aat_feature_group:n {Variant}
3369 \keys_define:nn {fontspec-aat}
3370 {
     Variant .default:n = {o} ,
3371
     Variant / unknown .code:n =
3372
3373
        \clist_map_inline:nn {#1}
3374
          { \@@_make_AAT_feature:nn {18}{##1} }
3375
3376
3377 }
3378 \aliasfontfeature{Variant}{StylisticSet}
3379 \00 define aat feature group:n {Vertical}
```

```
3380 keys_define:nn {fontspec-aat}
3381 {
3382
     Vertical .choice: ,
3383
     Vertical / RotatedGlyphs .code:n =
3384
3385
          \__fontspec_update_featstr:n {vertical}
3386
3387 }
3388
 38.8.10 Style
3389 \00 define aat feature group:n {Style}
                                       {Style} {Italic} {32} {2}
3390 \@@ define aat feature:nnnn
3391 \00 define aat feature:nnnn
                                       {Style} {Ruby} {28} {2}
3392 \@@_define_aat_feature:nnnn
                                      {Style} {Display} {19} {1}
3393 \@@_define_aat_feature:nnnn
                                      {Style} {Engraved} {19} {2}
                                       {Style} {TitlingCaps} {19} {4}
3394 \@@_define_aat_feature:nnnn
                                      {Style} {TallCaps} {19} {5}
3395 \@@_define_aat_feature:nnnn
 38.8.11 CJK shape
3396 \@@_define_aat_feature_group:n {CJKShape}
_{3397}\@0_define_aat_feature:nnnn
                                      {CJKShape} {Traditional} {20} {0}
                                      {CJKShape} {Simplified} {20} {1}
3398 \@@_define_aat_feature:nnnn
3399 \@@_define_aat_feature:nnnn
                                      {CJKShape} {JIS1978} {20} {2}
3400 \@@_define_aat_feature:nnnn
                                       {CJKShape} {JIS1983} {20} {3}
3401 \@@_define_aat_feature:nnnn
                                      {CJKShape} {JIS1990} {20} {4}
3402 \00 define aat feature:nnnn
                                      {CJKShape} {Expert} {20} {10}
                                      {CJKShape} {NLC} {20} {13}
3403 \@@ define aat feature:nnnn
 38.8.12 Character width
3404 \@@_define_aat_feature_group:n {CharacterWidth}
3405 \@@_define_aat_feature:nnnn
                                       {CharacterWidth} {Proportional} {22} {0}
3406 \@@_define_aat_feature:nnnn
                                       {CharacterWidth} {Full} {22} {1}
3407 \@@ define aat feature:nnnn
                                       {CharacterWidth} {Half} {22} {2}
3408 \@@_define_aat_feature:nnnn
                                       {CharacterWidth} {Third} {22} {3}
3409 \@@_define_aat_feature:nnnn
                                       {CharacterWidth} {Quarter} {22} {4}
_{3410}\ensuremath{\mbox{00\_define\_aat\_feature:nnnn}}
                                      {CharacterWidth} {AlternateProportional} {22} {5}
3411 \@@_define_aat_feature:nnnn
                                       {CharacterWidth} {AlternateHalf} {22} {6}
3412 \@@_define_aat_feature:nnnn
                                      {CharacterWidth} {Default} {22} {7}
 38.8.13 Annotation
3413 \@@_define_aat_feature_group:n {Annotation}
3414 \@@_define_aat_feature:nnnn
                                       {Annotation} {Off} {24} {0}
                                       {\rm Annotation} {\rm Box} {\rm 24} {\rm 1}
3415 \@@_define_aat_feature:nnnn
3416 \@@_define_aat_feature:nnnn
                                       {Annotation} {RoundedBox} {24} {2}
                                       {Annotation} {Circle} {24} {3}
3417 \@@_define_aat_feature:nnnn
                                       {Annotation} {BlackCircle} {24} {4}
3418 \@@_define_aat_feature:nnnn
3419 \@@_define_aat_feature:nnnn
                                       {Annotation} {Parenthesis} {24} {5}
                                       {Annotation} {Period} {24} {6}
3420 \@@_define_aat_feature:nnnn
                                       {Annotation} {RomanNumerals} {24} {7}
3421 \@@_define_aat_feature:nnnn
3422 \@@_define_aat_feature:nnnn
                                       {Annotation} {Diamond} {24} {8}
```

```
{Annotation} {BlackSquare} {24} {9}
3423 \@@_define_aat_feature:nnnn
                                      {Annotation} {BlackRoundSquare} {24} {10}
3424 \@@_define_aat_feature:nnnn
3425 \@@_define_aat_feature:nnnn
                                      {Annotation} {DoubleCircle} {24} {11}
```

Extended font encodings

To be removed after the 2017 release of LaTeX2e:

```
3426 \providecommand\UnicodeFontFile[2]{"[#1]:#2"}
                   3427 \providecommand\UnicodeFontName[2]{"#1:#2"}
                   3428 (xetexx)\providecommand\UnicodeFontTeXLigatures{mapping=tex-text;}
                   3429 (luatex)\providecommand\UnicodeFontTeXLigatures{+tlig;}
                   3430 \providecommand\add@unicode@accent[2]{#2\char#1\relax}
                   3431 \providecommand\DeclareUnicodeAccent[3] {%
                        \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
                   3433 }
 \EncodingCommand
                   3434 \DeclareDocumentCommand \EncodingCommand \mO{}m}
                   3435
                          \bool_if:NF \l_@@_defining_encoding_bool
                   3436
                             { \@@ error:nn {only-inside-encdef} \EncodingCommand }
                   3437
                   3438
                           \DeclareTextCommand{#1}{\UnicodeEncodingName}[#2]{#3}
                   3439
  \EncodingAccent
                   3440 \DeclareDocumentCommand \EncodingAccent {mm}
                   3441
                          \bool_if:NF \l_@@_defining_encoding_bool
                   3442
                             { \@@_error:nn {only-inside-encdef} \EncodingAccent }
                   3443
                          }
                   3445
  \EncodingSymbol
                   {\tt 3446} \setminus {\tt DeclareDocumentCommand} \setminus {\tt EncodingSymbol~\{mm\}}
                   3447
                          \bool_if:NF \l_@@_defining_encoding_bool
                   3448
                   3449
                             { \@@_error:nn {only-inside-encdef} \EncodingSymbol }
                   3450
                          \DeclareTextSymbol{#1}{\UnicodeEncodingName}{#2}
                   3451
\EncodingComposite
                   3452 \DeclareDocumentCommand \EncodingComposite {mmm}
                   3453
                          \bool if:NF \l @@ defining encoding bool
                   3454
                             { \@@ error:nn {only-inside-encdef} \EncodingComposite }
                   3455
                           \DeclareTextComposite{#1}{\UnicodeEncodingName}{#2}{#3}
                   3456
                        }
                   3457
```

\EncodingCompositeCommand

3458 \DeclareDocumentCommand \EncodingCompositeCommand {mmm}

```
3459
                                 \bool_if:NF \l_@@_defining_encoding_bool
                         3460
                         3461
                                   { \@@_error:nn {only-inside-encdef} \EncodingCompositeCommand }
                         3462
                                 \DeclareTextCompositeCommand{#1}{\UnicodeEncodingName}{#2}{#3}
                         3463
\DeclareUnicodeEncoding
                         3464 \DeclareDocumentCommand \DeclareUnicodeEncoding {mm}
                         3465
                         3466
                                 \DeclareFontEncoding{#1}{}{}
                                 \DeclareErrorFont{#1}{lmr}{m}{10}
                         3467
                                 \DeclareFontSubstitution{#1}{lmr}{m}{n}
                         3468
                                 \DeclareFontFamily{#1}{lmr}{}
                         3469
                         3470
                                 \DeclareFontShape{#1}{lmr}{m}{n}
                         3471
                                   {<->\UnicodeFontFile{lmroman10-regular}{\UnicodeFontTeXLigatures}}{}
                         3472
                                 \DeclareFontShape{#1}{lmr}{m}{it}
                         3473
                                   {<->\UnicodeFontFile{lmroman10-italic}{\UnicodeFontTeXLigatures}}{}
                         3474
                                 \DeclareFontShape{#1}{lmr}{m}{sc}
                         3475
                                   {<->\UnicodeFontFile{lmromancaps10-regular}{\UnicodeFontTeXLigatures}}{}
                         3476
                                 \DeclareFontShape{#1}{lmr}{bx}{n}
                         3477
                                   \label{limit} $$ <->\UnicodeFontFile{lmroman10-bold}_{\UnicodeFontTeXLigatures}$$ $$
                         3478
                                 \DeclareFontShape{#1}{lmr}{bx}{it}
                         3479
                                   {<->\UnicodeFontFile{lmroman10-bolditalic}{\UnicodeFontTeXLigatures}}{}
                         3480
                         3481
                         3482
                                 \tl_set_eq:NN \l_@0_prev_unicode_name_tl \UnicodeEncodingName
                         3483
                                 \tl_set:Nn \UnicodeEncodingName {#1}
                                 \bool_set_true: N \l_@@_defining_encoding_bool
                         3484
                         3485
                                 \bool_set_false:N \l_@@_defining_encoding_bool
                         3486
                                 \tl_set_eq:NN \UnicodeEncodingName \l_@@_prev_unicode_name_tl
                         3487
                         3488
       \UndeclareSymbol
                         3489 \DeclareDocumentCommand \UndeclareSymbol {m}
                         3490
                              {
                                 \bool_if:NF \l_@@_defining_encoding_bool
                         3491
                                   { \@@_error:nn {only-inside-encdef} \UndeclareSymbol }
                         3492
                                 \UndeclareTextCommand {#1} {\UnicodeEncodingName}
                         3493
                              }
                         3494
                         3495
    \UndeclareComposite
                         3496 \DeclareDocumentCommand \UndeclareComposite {mm}
                         3497
                                 \bool_if:NF \l_@@_defining_encoding_bool
                         3498
                                   { \@@_error:nn {only-inside-encdef} \UndeclareComposite }
                         3499
                                 \cs undefine:c
                         3500
                                   { \c_backslash_str \UnicodeEncodingName \token_to_str:N #1 - \tl_to_str:n {#2} }
                         3501
                         3502
```

40 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.

```
3503 \@ifpackageloaded{euler}
3504 {
                           \bool_set_true:N \g_@@_pkg_euler_loaded_bool
3505
3506
3507
                           \bool_set_false:N \g_@@_pkg_euler_loaded_bool
3508
3509
3510 \cs_set:Nn \fontspec_setup_maths:
3511 {
3512
                          \@ifpackageloaded{euler}
3513
                                    \bool_if:NTF \g_@@_pkg_euler_loaded_bool
3514
                                        { \begin{tabular}{ll} $\begin{tabular}{ll} $\begi
 3515
                                         { \@@_error:n {euler-too-late} }
 3516
                              }
3517
                              {}
3518
                           \@ifpackageloaded{lucbmath}{\bool_set_true:N \g_@@_math_lucida_bool}{}
 3519
                           \@ifpackageloaded{lucidabr}{\bool_set_true:N \g_@@_math_lucida_bool}{}
                           \@ifpackageloaded{lucimatx}{\bool set true:N \g @@ math lucida bool}{}
```

Knuth's CM fonts 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 LaTeX's operators maths font to still go back to the legacy cmr font for all these random glyphs, unless a separate maths font package has been loaded instead.

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

```
\label{legacymaths} $$ \DeclareSymbolFont{legacymaths}_{OT_1}_{cmr}_{m}_{n}$
      \SetSymbolFont{legacymaths}{bold}{OT1}{cmr}{bx}{n}
3523
      \DeclareMathAccent{\acute}
                                     {\mathalpha}{legacymaths}{19}
3524
     \DeclareMathAccent{\grave}
                                     {\mathalpha}{legacymaths}{18}
3525
                                     {\mathalpha}{legacymaths}{127}
     \DeclareMathAccent{\ddot}
3526
      \DeclareMathAccent{\tilde}
                                     {\mathalpha}{legacymaths}{126}
3527
     \DeclareMathAccent{\bar}
                                     {\mathalpha}{legacymaths}{22}
3528
     \DeclareMathAccent{\breve}
                                     {\mathalpha}{legacymaths}{21}
      \DeclareMathAccent{\check}
                                     {\mathalpha}{legacymaths}{20}
     \DeclareMathAccent{\hat}
                                     {\mathalpha}{legacymaths}{94} % too bad, eu-
   ler
```

```
3532 \DeclareMathAccent{\dot} {\mathalpha}{legacymaths}{95}
3533 \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{:}\mskip6muplus1mu\relax}
%
% euler.sty:
% \DeclareMathSymbol{:}\mathrel {EulerFraktur}{"3A}
%
% lucbmath.sty:
% \DeclareMathSymbol{\@tempb}{\mathpunct}{operators}{58}
% \ifx\colon\@tempb
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{58}
% \fi
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{58}
% \fi
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{58}
% \fi
% \DeclareMathSymbol{:}{\mathrel}{operators}{58}
```

 $(3A_16 = 58_10)$ 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.

```
3534 \group_begin:
3535 \mathchardef\@tempa="603A \relax
3536 \ifx\colon\@tempa
3537 \DeclareMathSymbol{\colon}{\mathpunct}{legacymaths}{58}
3538 \fi
3539 \group_end:
```

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

```
3540 \bool_if:NF \g_@@_math_euler_bool
3541 {
3542 \DeclareMathSymbol{!}{\mathclose}{legacymaths}{33}}
3543 \DeclareMathSymbol{:}{\mathrel} {legacymaths}{58}
3544 \DeclareMathSymbol{;}{\mathpunct}{legacymaths}{59}
3545 \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.

```
3546 \bool_if:NF \g_@@_math_lucida_bool
3547 {
3548 \DeclareMathSymbol{o}{\mathalpha}{legacymaths}{`o}
3549 \DeclareMathSymbol{1}{\mathalpha}{legacymaths}{`1}
3550 \DeclareMathSymbol{2}{\mathalpha}{legacymaths}{`2}
```

```
\DeclareMathSymbol{3}{\mathalpha}{legacymaths}{`3}
3551
3552
          \DeclareMathSymbol{4}{\mathalpha}{legacymaths}{`4}
          \DeclareMathSymbol{5}{\mathalpha}{legacymaths}{`5}
3553
          \DeclareMathSymbol{6}{\mathalpha}{legacymaths}{`6}
3554
          \DeclareMathSymbol{7}{\mathalpha}{legacymaths}{`7}
3555
          \DeclareMathSymbol{8}{\mathalpha}{legacymaths}{`8}
3556
          \DeclareMathSymbol{9}{\mathalpha}{legacymaths}{`9}
3557
          \DeclareMathSymbol{\Gamma}{\mathalpha}{legacymaths}{o}
3558
          \DeclareMathSymbol{\Delta}{\mathalpha}{legacymaths}{1}
3559
          \DeclareMathSymbol{\Theta}{\mathalpha}{legacymaths}{2}
3560
          \DeclareMathSymbol{\Lambda}{\mathalpha}{legacymaths}{3}
3561
          \DeclareMathSymbol{\Xi}{\mathalpha}{legacymaths}{4}
3562
          \DeclareMathSymbol{\Pi}{\mathalpha}{legacymaths}{5}
3563
          \DeclareMathSymbol{\Sigma}{\mathalpha}{legacymaths}{6}
3564
          \DeclareMathSymbol{\Upsilon}{\mathalpha}{legacymaths}{7}
3565
          \DeclareMathSymbol{\Phi}{\mathalpha}{legacymaths}{8}
3566
          \DeclareMathSymbol{\Psi}{\mathalpha}{legacymaths}{9}
3567
3568
          \DeclareMathSymbol{\Omega}{\mathalpha}{legacymaths}{10}
          \DeclareMathSymbol{+}{\mathbin}{legacymaths}{43}
3569
          \DeclareMathSymbol{=}{\mathrel}{legacymaths}{61}
3570
          \DeclareMathDelimiter{(}{\mathopen} {legacymaths}{40}{largesymbols}{0}
3571
          \DeclareMathDelimiter{)}{\mathclose}{legacymaths}{41}{largesymbols}{1}
3572
          \DeclareMathDelimiter{[]}\mathopen} {legacymaths}{91}{largesymbols}{2}
3573
          \DeclareMathDelimiter{]}{\mathclose}{legacymaths}{93}{largesymbols}{3}
3574
          \DeclareMathDelimiter{/}{\mathord}{legacymaths}{47}{largesymbols}{14}
3575
          \DeclareMathSymbol{\mathdollar}{\mathord}{legacymaths}{36}
3576
3577
3578
```

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

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

```
\DeclareSymbolFont{operators}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\updefault
     \SetSymbolFont{operators}{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\updefault
3580
     \DeclareSymbolFontAlphabet\mathrm{operators}
3581
     \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\itdefault
3582
     3583
     \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g_@@_mathsf_tl\mddefault\updefault
3584
     \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g_@@_mathtt_tl\mddefault\updefault
3585
     \SetSymbolFont{operators}{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\updefault
3586
     \tl_if_empty:NTF \g_@@_bfmathrm_tl
3587
3588
       \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\itdefault
3589
      }
3590
3591
       \SetMathAlphabet\mathrm{bold}\g fontspec encoding tl\g @@ bfmathrm tl\mddefault\updefault
3592
       \SetMathAlphabet\mathbf{bold}\g fontspec encoding tl\g @@ bfmathrm tl\bfdefault\updefault
3593
       \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mddefault\itdefault
3594
      }
3595
```

\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 'LTEXFont Catalogue': http://www.tug.dk/FontCatalogue/mathfonts.html. I'm sure there are more I've missed. Do the TEX 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.

```
3599 \cs_new: Nn \fontspec_maybe_setup_maths:
3600 {
      \@ifpackageloaded{anttor}
3601
3602
        \ifx\define@antt@mathversions a\bool_set_false:N \g_@@_math_bool\fi
3603
3604
      \@ifpackageloaded{arevmath}{\bool_set_false:N \g_@@_math_bool}{}
3605
      \@ifpackageloaded{eulervm}{\bool_set_false:N \g_@@_math_bool}{}
3606
      \@ifpackageloaded{mathdesign}{\bool_set_false:N \g_@@_math_bool}{}
      \@ifpackageloaded{concmath}{\bool_set_false:N \g_@@_math_bool}{}
3609
      \@ifpackageloaded{cmbright}{\bool_set_false:N \g_@@_math_bool}{}
3610
      \@ifpackageloaded{mathesf}{\bool set false:N \g @@ math bool}{}
      \@ifpackageloaded{gfsartemisia}{\bool set false:N \g @@ math bool}{}
3611
      \@ifpackageloaded{gfsneohellenic}{\bool_set_false:N \g_@@_math_bool}{}
3612
      \@ifpackageloaded{iwona}
3613
3614
       \ifx\define@iwona@mathversions a\bool_set_false:N \g_@@_math_bool\fi
3615
3616
      \@ifpackageloaded{kpfonts}{\bool_set_false:N \g_@@_math_bool}{}
3617
3618
      \@ifpackageloaded{kmath}{\bool_set_false:N \g_@@_math_bool}{}
      \@ifpackageloaded{kurier}
3619
3620
       \ifx\define@kurier@mathversions a\bool_set_false:N \g_@@_math_bool\fi
3621
3622
      \@ifpackageloaded{fouriernc}{\bool_set_false:N \g_@@_math_bool}{}
3623
      \@ifpackageloaded{fourier}{\bool set false:N \g @@ math bool}{}
3624
      \@ifpackageloaded{lmodern}{\bool_set_false:N \g_@@_math_bool}{}
3625
      \@ifpackageloaded{mathpazo}{\bool_set_false:N \g_@@_math_bool}{}
3626
      \label{local_condition} $$ \end{mathptmx}_{\bool_set_false:N \g_00_math_bool}_{\end{mathptmx}}$$
3627
      \@ifpackageloaded{MinionPro}{\bool_set_false:N \g_@@_math_bool}{}
      \@ifpackageloaded{unicode-math}{\bool_set_false:N \g_@@_math_bool}{}
3629
      \@ifpackageloaded{breqn}{\bool set false:N \g @@ math bool}{}
3630
      \bool if:NT \g @@ math bool
3631
      {
3632
        \@@ info:n {setup-math}
3633
       \fontspec_setup_maths:
3634
      }
3635
3636 }
3637 \AtBeginDocument{\fontspec_maybe_setup_maths:}
```

41 Closing code

41.1 Compatibility

\zf@enc Old interfaces. These are needed by, at least, the mathspec package.

41.2 Finishing up

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

42 Patching code

3651 (*fontspec)

42.1 Italic small caps and so on

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

```
3652 \providecommand*\itscdefault{\itdefault\scdefault}
3653 \providecommand*\slscdefault{\sldefault\scdefault}
3654 \DeclareRobustCommand{\sishape}
3655 {
3656 \not@math@alphabet\sishape\relax
3657 \fontshape{\itscdefault}\selectfont
3658 }
3659 \DeclareTextFontCommand{\textsi}{\sishape}
```

Lagrange of the Standard Stand

```
3660 \cs_new:Nn \@@_shape_merge:nn { c_@@_shape_#1_#2_tl }
3661 \tl_const:cn { \@@_shape_merge:nn \itdefault \scdefault } {\\itscdefault}
3662 \tl_const:cn { \@@_shape_merge:nn \scdefault \itdefault } {\\itscdefault}
3663 \tl_const:cn { \@@_shape_merge:nn \scdefault \itdefault } {\\itscdefault}
3664 \tl_const:cn { \@@_shape_merge:nn \scdefault \itdefault } {\\itscdefault}
3665 \tl_const:cn { \@@_shape_merge:nn \slscdefault \itdefault } {\\itscdefault}
3666 \tl_const:cn { \@@_shape_merge:nn \itscdefault \sldefault } {\\itscdefault}
```

```
3667 \tl_const:cn { \@@_shape_merge:nn \itscdefault \updefault } {\scdefault}
                                                            3668 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.
                                                            3669 \cs_new: Nn \fontspec_merge_shape:n
                                                            3670 {
                                                            3671
                                                                        \bool if:nTF
                                                            3672
                                                                                    \tl if exist p:c { \@@ shape merge:nn {\f@shape} {#1} }
                                                            3673
                                                                                    \cs_if_exist_p:c
                                                            3674
                                                            3675
                                                                                              \f@encoding/\f@family/\f@series/
                                                            3676
                                                                                              \tl_use:c { \00_shape_merge:nn {\f0shape} {#1} }
                                                            3677
                                                            3678
                                                            3679
                                                                               { \int \int \int d^2 x d^
                                                                    lectfont }
                                                                              { \fontshape {#1} \selectfont }
                                                            3681
                                                            3682 }
                                      \itshape The original \...shape commands are redefined to use the merge shape macro.
                                      \scshape 3683 \DeclareRobustCommand \itshape
                                      \upshape 3684 {
                                                                          \not@math@alphabet\itshape\mathit
                                      \slshape 3685
                                                           3686
                                                                        \fontspec_merge_shape:n\itdefault
                                                           3687 }
                                                            3688 \DeclareRobustCommand \slshape
                                                            3689 {
                                                                          \not@math@alphabet\slshape\relax
                                                                         \fontspec_merge_shape:n\sldefault
                                                            3692 }
                                                            3693 \DeclareRobustCommand \scshape
                                                            3694 {
                                                                          \not@math@alphabet\scshape\relax
                                                            3695
                                                                         \fontspec_merge_shape:n\scdefault
                                                            3696
                                                            3697 }
                                                           3698 \DeclareRobustCommand \upshape
                                                                        \not@math@alphabet\upshape\relax
                                                                       \fontspec_merge_shape:n\updefault
                                                            3702 }
                                                                                 Emphasis
                                                  \em Redefinition of \{\mbox{\em } \dots\} and \mbox{\em } nesting of emphases.
                                             \label{lem:monopole} $$ \operatorname{protected:Npn \emfontdeclare} $$ $$ $$ $$ $$ $$ $$ $$ $$
                                      \verb|\emshape|_{3704} \quad \{
```

\prop_clear:N \g_@@_em_prop

\group begin:

\int zero:N \l @@ emdef int

\eminnershape 3705

\emfontdeclare 3706

3707

```
\normalfont
  3708
  3709
             \clist_map_inline:nn {\emreset,#1}
  3710
               {
  3711
                 ##1
                 \prop_gput:NxV \g_00_em_prop { \f0series/\f0shape } { \l_00_emdef_int }
  3712
                 \prop_gput:Nxn \g_@@_em_prop { switch-\int_use:N \l_@@_emdef_int } { ##1 }
  3713
                 \int_incr:N \l_@@_emdef_int
  3714
  3715
           \group_end:
  3716
        }
  3717
  3718 \DeclareRobustCommand \em
  3719
        {
           \@nomath\em
  3720
  3721
           \prop_get:NxNT \g_@0_em_prop { \f@series/\f@shape } \l_@0_em_tmp_tl
  3722
             { \int_set:Nn \l_@@_em_int { \l_@@_em_tmp_tl } }
  3723
  3724
           \int_incr:N \l_@@_em_int
  3725
  3726
           \prop get:NxNTF \g @@ em prop { switch-\int use:N \l @@ em int } \l @@ em switch tl
  3727
             { \label{local_em_switch_tl} }
  3728
  3729
               \int_zero:N \l_@@_em_int
  3730
               \emreset
  3731
  3732
  3733
  3734
    Document commands:
  3735 \DeclareTextFontCommand{\emph}{\em}
  3736 \cs_set:Npn \emreset { \upshape }
  3737 \cs_set:Npn \emshape { \itshape }
  3738\cs_set:Npn \eminnershape { \upshape }
  3739 \emfontdeclare{ \emshape, \eminnershape }
    Ensure nesting resets when necessary:
  3740 \cs_set:Npn \reset@font { \normalfont \int_zero:N \l__fontspec_em_int }
    Programmer's interface for setting nesting level:
  3741 \cs_new: Nn \fontspec_set_em_level:n { \int_set: Nn \l_@@_em_int {#1} }
    42.3 \-
\- This macro is courtesy of Frank Mittelbach and the \LaTeX 2\varepsilon source code.
  3742 \DeclareRobustCommand{\-}
  3743 {
        \discretionary
  3744
  3745
           \char\ifnum\hyphenchar\font<\z@
  3746
                  \xlx@defaulthyphenchar
  3747
  3748
                  \hyphenchar\font
  3749
```

```
3750 \fi
3751 }{}{}
3752 }
3753 \def\xlx@defaulthyphenchar{`\-}
```

42.4 Verbatims

3784 {

3785 \fontspec patch verbatim:

Many thanks to Apostolos Syropoulos for discovering this problem and writing the redefinion 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.
                               3754 \cs_new: Nn \fontspec_visible_space:
                               3755 {
                               3756
                                   \@@_primitive_font_glyph_if_exist:NnTF \font {"2423}
                                    { \char"2423\scan_stop: }
                                     { \fontspec_visible_space_fallback: }
                              3758
                               3759 }
tspec_visible_space_fallback: If the current font doesn't have u+2423: OPEN BOX, use Latin Modern Mono instead.
                               3760 \cs_new: Nn \fontspec_visible_space_fallback:
                               3761 {
                               3762
                               3763
                                      \usefont{\g_fontspec_encoding_tl}{lmtt}{\f@series}{\f@shape}
                               3764
                                     \textvisiblespace
                              3765
                              3766 }
ontspec_print_visible_spaces: Helper macro to turn spaces (^^20) active and print visible space instead.
                              3767 \group_begin:
                               3768 \char_set_catcode_active:n{"20}%
                              3769 \cs_gset: Npn\fontspec_print_visible_spaces: {%
                               3770 \char_set_catcode_active:n{"20}%
                               3771 \cs_set_eq:NN^^20\fontspec_visible_space:%
                               3772 }%
                              3773 \group_end:
                        \verb Redefine \verb to use \fontspec_print_visible_spaces:.
                        \verb* 3774 \def\verb
                              3775 {
                                    \relax\ifmmode\hbox\else\leavevmode\null\fi
                              3776
                              3777
                                       \verb@eol@error \let\do\@makeother \dospecials
                              3778
                                       \verbatim@font\@noligs
                               3779
                                      \@ifstar\@@sverb\@verb
                              3780
                              3781 }
                               3782 \def\@@sverb{\fontspec_print_visible_spaces:\@sverb}
                                   It's better to put small things into \AtBeginDocument, so here we go:
                               3783 \AtBeginDocument
```

```
\fontspec_patch_moreverb:
             3786
             3787
                   \fontspec_patch_fancyvrb:
             3788
                   \fontspec_patch_listings:
             3789 }
   verbatim* With the verbatim package.
             3790 \cs_set:Npn \fontspec_patch_verbatim:
                   \@ifpackageloaded{verbatim}
             3792
             3793
                      \cs set:cpn {verbatim*}
             3794
             3795
                        \group_begin: \@verbatim \fontspec_print_visible_spaces: \verbatim@start
             3796
             3797
                     }
             3798
              This is for vanilla LATEX.
             3799
                      \cs_set:cpn {verbatim*}
             3800
             3801
             3802
                        \@verbatim \fontspec_print_visible_spaces: \@sxverbatim
             3803
                     }
             3804
             3805 }
listingcont*
              This is for moreverb. The main listing* environment inherits this definition.
             3806 \cs_set:Npn \fontspec_patch_moreverb:
             3807 {
                   \@ifpackageloaded{moreverb}{
             3808
                      \cs_set:cpn {listingcont*}
             3809
             3810
                       {
                        \cs set:Npn \verbatim@processline
             3811
             3812
                         {
                          \thelisting@line \global\advance\listing@line\c one
             3813
                          \the\verbatim@line\par
             3814
                         }
             3815
                        \@verbatim \fontspec_print_visible_spaces: \verbatim@start
             3816
             3817
             3818
                   }{}
             3819 }
                  listings and fancvrb make things nice and easy:
             3820 \cs_set:Npn \fontspec_patch_fancyvrb:
             3821 {
                   \@ifpackageloaded{fancyvrb}
             3822
             3823
                      \cs_set_eq:NN \FancyVerbSpace \fontspec_visible_space:
             3824
             3825
                    }{}
             3826 }
             3827 \cs_set:Npn \fontspec_patch_listings:
             3828 {
```

42.5 \oldstylenums

\oldstylenums This comm \liningnums command.

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

```
3834 \RenewDocumentCommand \oldstylenums {m}
3835 {
3836     { \addfontfeature{Numbers=OldStyle} #1 }
3837 }
3838 \NewDocumentCommand \liningnums {m}
3839 {
3840      { \addfontfeature{Numbers=Lining} #1 }
3841 }
3842 \( \fontspec \rangle \)
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	\@@_define_aat_f
\#	625, <u>18</u>
1586, 2478-2482	3332, 3343
\ 1539, 3742	3368, 3379
\@@_DeclareFontShape:nnnnnn 1409	\@@_define_opent
\@@_DeclareFontShape:xxxxxx	\@@_define_opent
\@@_add_nfssfont:nnnn	1596-1598
1210–1215, 1230, 2174	2499, 2512
\@@_aff_error:n 1923, 2222, 2263, 2302	2574, 2580
\@@_calc_scale:n 2181, 2182, 2187	2591, 2594
\@@_check_lang:n 1663	\@@_define_opent
\@@_check_lang:nTF	
=	2498, 2511
827, 844, 1663, 2772, 2794, 2801	2573, 2583
\@@_check_ot_feat:n 1696	2657, 2666
\00_check_ot_feat:nF 1615	\@@_define_opent
\@@_check_ot_feat:nTF	
772, 793, 1491, 1696	2504-2510
\@@_check_script:n 1634	2539-2544
\@@_check_script:nTF	2572, 2579
808, 1081, <u>1634,</u> 2738, 2755	2631-2638
\@@_combo_sc_shape:n	\@@_define_opent
1399, 1403, 1448, 1456	
\@@_construct_font_call:nn	\@@_error:n
903, 1041, 1043, 1046, 1048,	\@@_error:nn
1051, 1179, 1295, 1296, 1316, 1385	3443, 3449
\@@_construct_font_call:nnnnnn	\@@_error:nx
	. 126, 1044
\@@_declare_shape:nnnn <u>1319</u>	\@@_extract_all_
\@@_declare_shape:nnxx 1306	\@@_extract_all_
<pre>\@@_declare_shape_loginfo:nn</pre>	\@@_feat_off:n .
	\@@_feat_prop_ad
\@@_declare_shape_slanted:nn	\@@_feat_reset:n
	\@@_find_autofon
\@@_declare_shapes_normal:nn	
	\@@_font_is_file
\@@_declare_shapes_smcaps:nn	\@@_font_is_file
1329, 1394	\@@_ft :=
\@@_define_aat_feature:nnnn	\@@_font_is_name
628, 1863, 3290-	\@@_font_suppres
3309, 3318-3322, 3324-3331,	\@@ fontnamo ura
3333-3342, 3344-3346, 3348-	\@@_fontname_wra
3351, 3353-3355, 3390-3395,	
3397-3403, 3405-3412, 3414-3425	\@@_get_features

```
\ \ @@ dof:
                 feature_group:n ..
                 <u>861</u>, 3289, 3317, 3323,
                 .3, 3347, 3352, 3356,
                 9, 3389, 3396, 3404, 3413
                 type_feature:nnn 636
                 type_feature:nnnnn
                  .....<u>1574</u>,
                 8, 1602, 1603, 2484,
                 2, 2530, 2546, 2562,
                 0-2582, 2584, 2589-
                 4, 2617, 2618, 2620, 2640
                 type_feature_group:n
                  . . . 633, 1570, 2483,
                 1, 2529, 2545, 2561,
                 3, 2593, 2619, 2639,
                 6, 2679, 2692, 2713, 2722
                 type_onoffreset:nnnnn
                  . . 1592, 2490-2495,
                 0, 2521-2525, 2528,
                 4, 2555–2560, 2569–
                 19, 2592, 2607–2616,
                 8, 2650–2656, 2658–2665
                 type_onreset:nnnnn
                  . . . . . <u>1600</u>, 2496, 2497
                 . . . . . . . 124, 1341, 3516
                 ..... 125, 3437,
                 9, 3455, 3461, 3492, 3499
                 4, 1317, 1926, 2275, 2299
                 _features: .... 1000
                 _features:n 931,1000
                 . . . . . . . . . . 1592, 1597
                 dd:nn .......
                   1574, 1586, 2478-2482
                 n . . . . 1593, 1598, 1603
                 onts: ..... 1167,1187
                 e: ..... 902, 1934
                 e:,\@@_font_is_name:
                  .....<u>1066</u>
                 e: ..... 1528
                 ss_not_found_error:
                  . . . . 425, 440, 700, 918
                 ap:n ......
                  . 1053, 1068, 1072, 1073
                 s:Nn 706,939,1105,1379
```

\@@_head_ii:n <u>100</u>	211, 217, 221, 231, 235, 240, 244,
\@@_head_ii:w 100, 101	249, 254, 259, 265, 269, 276, 280,
\@@_if_autofont:nn 1293	284, 288, 293, 297, 302, 307, 311,
\@@_if_autofont:nnTF 1285	319, 323, 327, 331, 335, 340, 345
\@@_if_detect_external:n 967	\@@_msg_new:nnnn
\@@_if_detect_external:nT 902,925,967	138, 155, 164, 175, 185, 193, 201
\@@_if_font_feature:n 696	\@@_ot_compat:nn 2814, 2818-2834
\@@_if_font_feature:nTF 694	\@@_preparse_features: 936, 1024
\@@_info:n 130, 1092, 2198, 3633	\@@_primitive_font_glyph_if_exist:Nn
\@@_info:nx 131,1287	
\@@_info:nxx 132,1170	\@@_primitive_font_glyph_if_exist:NnTF
\@@_init: 701, 901, 919, 1523	
\@@_init_fontface: 1108, 1545	\@@_primitive_font_gset:Nnn 1047
\@@_init_ttc:n 928, 979	\@@_primitive_font_if_exist:n 437
\@@_int_mult_truncate:Nn 103, 2336	\@@_primitive_font_if_exist:nTF 903
\@@_iv_str_to_num:\\n\ 791,	\@@_primitive_font_if_null:N 429
792, 843, 1555, 1640, 1669, 1709	<pre>\@@_primitive_font_if_null:NT</pre>
	1044, 1317
\@@_iv_str_to_num:w 1557, 1559	\@@_primitive_font_if_null:NTF . 442
\@@_keys_define_code:nnn	\@@_primitive_font_set:Nnn . 441,
1919, 1925, 1928, 1940,	727, 746, 759, 787, 804, 819, 838,
1941, 1950, 1984, 1989, 1994, 2001, 2006, 2011, 2015, 2019,	855, 872, 1042, 1295, 1296, 1316
2044, 2055, 2059, 2063, 2067,	\@@_primitive_font_set:Nnn,\@@_primitive_font_gset:Nnn
2078, 2082, 2089, 2093, 2097,	
2101, 2105, 2112, 2117, 2123,	\@@_remove_clashing_featstr:n
2127, 2131, 2135, 2139, 2143,	<u>1514</u> , 1624
2147, 2154, 2177, 2217, 2243,	\@@_sanitise_fontname:Nn
2264, 2268, 2272, 2303, 2333,	. 571, 921–923, <u>953</u> , 983–985, 991
2348, 2365, 2369, 2373, 2474, 2784	\@@_save_family:nn 944, <u>1164</u>
\@@_keys_set_known:nnN 93	\00_save_family_needed:n 1135
\@@_keys_set_known:nxN 1028,	\00_save_family_needed:nTF 942, 1135
1033, 1035, 1109, 1111, 1236, 1304	\@@_save_fontinfo:n 1166,1172
\@@_load_external_fontoptions:Nn	\@@_select_font_family:nn
	. 597, 600, 887, 894, <u>914</u> , 952, 3641
\@@_load_font: 937, 1038	\@@_set_autofont:Nnn
\@@_load_fontname:n	1191–1193, 1198, 1203, 1206, <u>1277</u>
1305, 1310, 1345, 1366	\@@_set_default_features:nn 556,560
\@@_make_AAT_feature:nn	\@@_set_faces: 1169, <u>1208</u>
1867, 1870, 3364, 3375	\@@_set_faces_aux:nnnnn 1217,1219
\@@_make_AAT_feature_string:nn 1882	\@@_set_font_default_features:nnn
\@@_make_AAT_feature_string:nnTF	557, 565
	\@@_set_font_dimen:NnN
\@@ make OT feature:nnn 1589,	2191, 2192, <u>2201</u>
1605, 1630, 2671, 2675, 2719, 2728	\@@_set_font_type: . 747,760,788,
\@@_make_OT_feature:xxx 2687,2698	805, 820, 839, 856, 873, 1045, <u>1244</u>
\@@_make_font_shapes:Nnnnn 1222, 1301	\@@_set_scriptlang: 938, <u>1075</u>
\@@_make_ot_smallcaps:TF 1488	\@@_setup_nfss:Nnnn . 1346, 1371, <u>1375</u>
\@@_make_smallcaps:TF 1356, 1488, 1494	\@@_setup_single_size:nn . 1326, 1334
\@@_msg_new:nnn	\@@_shape_merge:nn 1405,
136, 141, 146, 150, 171,	1406, 3660–3668, 3673, 3677, 3680

Recomposition Recompositio	\@@_swap_plus_minus:n 1625, 1631	\AtBeginDocument 405, 3637, 3783
\[\text{\qquad} \] \text{\qquad} \] \text{\qquad} \] \text{\qquad} \] \text{\qquad} \] \text{\qquad} \] \qquad} \[\text{\qqquad} \] \qquad} \qquad} \] \qquad} \qquad} \] \qquad} \qquad} \] \qquad} \qqquad} \qquad} \qqquad} \qqquad} \qqquad} \qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq		
\text{Vector} Vector		
1215, 1439, 1442, 1443, 1451, 1457, 2370, 2351, 2367, 2371, 2383, 2402, 2410, 2419, 2476, 3314, 262 warning:n 127, 356, 381, 387, 1873, 1997, 2358, 2362, 2389, 2427 @_warning:nx		
1877, 2270, 2351, 2367, 2371, 2363, 2402, 2410, 2419, 2476, 3314, 2383, 2402, 2410, 2419, 2476, 3314, 2383, 2402, 2410, 2419, 2476, 3314, 2383, 2402, 2439, 2427, 2438, 2430, 2752, 2757, 2762, 2779, 2808 @@_warning:nx		-
388, 3586, 3589, 3593, 3596, 3597 \text{Veq_varning:n} \		
\text{		
1873, 1997, 2358, 2362, 2389, 2427		
1201, 1279, 1352, 1369, 1507,		_ = =
1944, 2107, 2219, 2316, 2327, 2318, 2344, 2340, 2752, 2757, 2762, 2779, 2808		
1879, 1963, 1968, 2317, 2328, 2340, 2752, 2757, 2762, 2779, 2808 2340, 2752, 2757, 2762, 2779, 2808 200_warning:nxx		
340, 2752, 2757, 2762, 2779, 2808 @@_warning:nxx		
\(\begin{arrangmatrix} \ 129, 627, 635, 945 \\ \begin{arrangmatrix} \ 3780, 3782 \\ \emptyred{clibrackageloaded} \ \ \ 3503, 3512, 3519-3521, \ 3601, 3605-3613, 3617-3619, \ 3623-3630, 3792, 3808, 3822, 3829 \\ \emptyred{clibrackageloaded} \ \ 3780 \\ \emptyred{cmakeother} \ 37780 \\ \emptyred{cmakeother} \ 37780 \\ \emptyred{cmakeother} \ 37780 \\ \emptyred{cmakeother} \ 3782 \\ \emptyred{conmath} \ 3720 \\ \emptyred{conmath} \ 3720 \\ \emptyred{conmath} \ 3780 \\ \emptyred{csverb} \ 3782 \\ \emptyred{csverb} \ 3782 \\ \emptyred{csverb} \ 3783 \\ \emptyred{csverb} \ 3786 \\		
\(\begin{array}{c} \lambda \text{ \ \ \text{Qfilelist} \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \		
1468, 1499, 1613, 1622, 2274, 2381, 2388, 2434, 2456, 3631, 3645		
2381, 2388, 2434, 2456, 3631, 3645		
\(\) \(\)		
\langle \lan		
748,761,789,806,821,840,857, 874,977,1003,1113,1265,1496, 1636,1653,1659,1665,1683, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514, 1692,1698,1722,1735,2376,3514,261,261,274,2348,3508,3603,3605-3612,3615,3617,3618,3621,3623-3630, 1692,1610,1647,1673,1711,2026,2051,2074,2439,2461,2735,3486,3508,3603,3605-3612,3615,3617,3618,3621,3623-3630, 1692,1610,1647,1673,1716,1930-1932,2023,2048,2071,2742,3484,3505,3515,3519-3521,2612,262,2223,223,2244,23444,236161618,1648,1674,1712,262,2243,2444,2361618,242,242,2434,244,2436161618,1648,1648,1674,1712,266,2618,2618,2618,2618,2618,2618,261		
\\(\) \(\) \(\) \\ \) \(\) \\ \) \\ \\ \) \\ \\ \\ \) \\ \\ \\		
\(\) \(\)		
1692, 1698, 1722, 1735, 2376, 3514	37	
\langle bool_internstr		" " " " " " " " " " " " " " " " " " "
\langle \lan		
\(\lambda\) \(\sigma\)		
\(\circ\) \(\circ\) \(\sigma\) \(\sigma\) \(\sigma\) \(\circ\) \(\circ\) \(\circ\) \(\sigma\) \(\circ\) \(
\(\)\(\)\(\)\(\)\(\)\(\)\(\)\(\)\(\)\(\		T
\(\text{Qverb} \) \(\text{3535}, \text{3536} \) \(\text{Qverb} \) \(\text{3796}, \text{3802}, \text{3816} \) \(\text{10} \) \(\text{11}, \text{152}, \text{160}, \text{161}, \text{224-226}, \) \(\text{237}, \text{273}, \text{299}, \text{299}, \text{304}, \text{3145-316}, \) \(\text{337}, \text{348-350}, \text{1435}, \text{1447}, \text{1461} \) \(\text{2dim} \text{eval} \text{eval} \) \(\text{2220}, \text{2223} \) \(\text{2360} \) \(\text{2360}, \text{1363}, \text{1577}, \text{1256}, \) \(\text{1258}, \text{1261}, \text{1274}, \text{1363}, \text{1527}, \) \(\text{1609}, \text{1610}, \text{1647}, \text{1716}, \) \(\text{1930-1932}, \text{2023}, \text{2048}, \text{2071}, \) \(\text{2742}, \text{3484}, \text{3505}, \text{3515}, \text{3519-3521} \) \(\text{bool} \text{until} \downarrow \text{dormal} \) \(\text{2220}, \text{2223} \) \(\text{2223} \) \(\text{2200}, \text{233} \) \(\text{2360} \) \(\text{2360} \) \(\text{2360}, \text{2361}, \text{3678}, \text{3610}, \text{3618}, \text{3617}, \text{363}, \text{3659}, \text{3665}, \text{3687}, \text{3618}, \text{3621}, \text{3623-3630} \) \(\text{bool} \) \(\text{256}, \text{1256}, \text{1256}, \text{1256}, \text{1256}, \text{1256}, \text{1258}, \text{1261}, \text{1274}, \text{1363}, \text{1527}, \text{1609}, \text{1610}, \text{1647}, \text{1677}, \text{1716}, \text{1930-1932}, \text{2023}, \text{2048}, \text{2071}, \text{2742}, \text{3484}, \text{3505}, \text{3515}, \text{3519-3521} \) \(\text{bool} \) \(\text{until} \downarrow \text{360}, \text{360}, \text{360}, \text{3616}, \text{3617}, \text{3617}, \text{3617}, \text{363}, \text{1256}, \text{1256}, \text{1256}, \text{1256}, \text{1256}, \text{1256}, \text{1256}, \text{1256}, \text{1257}, \text{242}, \text{3484}, \text{3505}, \text{3515}, \text{3519-3521} \) \(\text{bool} \) \(\text{1600} \) \(\te		
\(\congruence{\congruen		
\[\ldots \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \		
3615, 3617, 3618, 3621, 3623-3630		
\langle		
237, 273, 290, 299, 304, 314-316, 337, 342, 348-350, 1435, 1447, 1461 \dim_eval:w		
337, 342, 348-350, 1435, 1447, 1461 _dim_eval:w		
\dim_eval_end:		
_fontspec_update_featstr:n . 3385 _fontspec_parse_wordspace:w		1258, 1261, 1274, 1363, 1527,
_fontspec_parse_wordspace:w 2220, 2223 \\ \text{ bool_until_do:nn 1644, 1674, 1712} \\ \text{ breve 3529} \\ \add@unicode@accent .3430, 3432, 3444 \\ \addfontfeature 223, 611, 3836, 3840 \\ \addfontfeatures 203, \frac{587}{287} \\ \advance 3813 \\ \text{ c_empty_tl 1557, 1565, 1566} \\ \text{ c_one 427, 3813} \\ \text{ c_zero 1257, 2294} \\ \aliasfontfeatureoption \\ \text{ c_thar 3430, 3746, 3757}		1609, 1610, 1647, 1677, 1716,
A \acute		1930-1932, 2023, 2048, 2071,
A \acute		2742, 3484, 3505, 3515, 3519–3521
\acute	2220, <u>2223</u>	
\acute	Α	\breve 3529
\add@unicode@accent 3430, 3432, 3444 \c_backslash_str 3501 \addfontfeature 223, 611, 3836, 3840 \c_colon_str 2696, 2708 \addfontfeatures 203, 587 \c_empty_tl 1557, 1565, 1566 \c_minus_one 2280 \c_one 427, 3813 \c_sol_ene 2332, 2678, 2691, 3378 \c_zero 1257, 2294 \char 3430, 3746, 3757		C
\addfontfeature . 223, 611, 3836, 3840 \c_colon_str		\ a haalaalaah ata
\addfontfeatures		
\advance		
\aliasfontfeature		
. <u>639</u> , 1939, 2332, 2678, 2691, 3378 \c_zero		
\aliasfontfeatureoption \char 3430, 3746, 3757		
		\text{\tin}\text{\tex{\tex

	I
\char_set_catcode_ignore:n 353	\cs_set:cpn 3794, 3800, 3809
\char_set_catcode_space:n 140	\cs_set:Nn 1051, 1056, 1105,
\check 3530	1224, 1488, 1489, 1494, 1555, 3510
\clist_clear:N 705, 1009, 1012, 1315	\cs_set:Npn 100, 101, 413, 417,
\clist_count:N 2159	421, 425, 1072, 1073, 1523, 1559,
\clist_count:n 227	2223, 2322, 3639, 3736–3738,
\clist_map_break: 963, 975, 2744	3740, 3790, 3806, 3811, 3820, 3827
\clist_map_inline:\n 663, 957, 970	\cs_set_eq:cN 1143
	\cs_set_eq:NN 412, 415, 509, 611, 638,
\clist_map_inline:nn	907, 952, 1068, 3771, 3824, 3831
567, 643, 1326, 1517, 2674,	\cs_to_str:\N\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
2685, 2705, 2736, 3362, 3374, 3709	
\clist_new:N 54, 55	\cs_undefine:c . 1414, 2150, 2152, 3500
\clist_put_left:Nn 1359	\cyrillicencoding 403,407
\clist_put_right:Nn	D
562, 2115, 2437, 2459	D
\clist_put_right:Nx	\ddot
2443, 2445, 2465, 2467	\DeclareDocumentCommand . 461,477,
\clist_set:Nn 56, 562, 686, 1235,	493, 510, 515, 520, 525, 536, 549,
1465, 1913, 2080, 2084, 2091,	553, 587, 612, 622, 630, 639, 659,
2095, 2099, 2103, 2109, 2114, 2119	676, 680, 684, 690, 3434, 3440,
\clist_set:Nx 1005, 1015	3446, 3452, 3458, 3464, 3489, 3496
\clist_set_eq:NN 1350	\DeclareErrorFont 3467
\colon 3536, 3537	\DeclareFontEncoding 382,3466
\color@ 2305	\DeclareFontFamily 1168,3469
\convertcolorspec 2307	\DeclareFontsExtensions 684
\cs:w 570	\DeclareFontShape
	1416, 3471, 3473, 3475, 3477, 3479
\cs_end: 570	\DeclareFontSubstitution 383,3468
\cs_generate_variant:Nn	\DeclareMathAccent 3524-3533
99, 102, 107–121, 134,	\DeclareMathDelimiter 3571-3575
135, 890, 1333, 1418, 1569, 1630	\DeclareMathSymbol
\cs_gset:Npn 3769	3537, 3542–3545, 3548–3570, 3576
\cs_if_exist:cF 379, 1145	\DeclareOption 355, 357-363, 368
\cs_if_exist:cTF 720, 1151, 1161, 2305	\DeclareRobustCommand
\cs_if_exist:NT 1141	465, 481, 497, 541, 3654,
\cs_if_exist_p:c 3674	3683, 3688, 3693, 3698, 3718, 3742
\cs_new:Nn	\DeclareSymbolFont 3522,3579
93, 103, 136, 138, 560, 565, 884,	
891, 953, 979, 988, 1000, 1024,	\DeclareSymbolFontAlphabet 3581
1038, 1066, 1070, 1075, 1164,	\DeclareTextCommand . 3432, 3438, 3444
1172, 1187, 1208, 1219, 1230,	\DeclareTextComposite 3456
1244, 1277, 1301, 1310, 1319,	\DeclareTextCompositeCommand 3462
1334, 1375, 1389, 1394, 1403,	\DeclareTextFontCommand 3659, 3735
1409, 1419, 1431, 1504, 1514,	\DeclareTextSymbol 3450
1545, 1570, 1574, 1584, 1592-	\DeclareUnicodeAccent 3431
1594, 1600, 1605, 1631, 1632,	\DeclareUnicodeEncoding 144, 3464
1861, 1863, 1870, 1919, 1923,	\def 3753, 3774, 3782
2187, 2201, 2732, 2769, 2814,	\defaultfontfeatures 553
3599, 3660, 3669, 3741, 3754, 3760	\define@antt@mathversions 3603
\cs_new:Npn 23, 24, 124-133, 2695	\define@iwona@mathversions 3615
,	
	1
\cs_new_protected:Npn 3703	\Delta 3559

	1
\dim_compare:nNnT 2204	766, 769, 770, 786, 803, 818, 823,
\dim_new:N 50-52	825, 837, 854, 859, 871, 876, 3676
\dim_set:Nn 2203	\f@series 3676, 3712, 3722, 3763
\dim_to_fp:n 2195, 2196	\f@shape
\directlua 5, 1658, 1688, 1728	3673, 3677, 3680, 3712, 3722, 3763
\discretionary 3744	\f@size 727,746,
\do 3778	759, 787, 804, 819, 838, 855, 872,
\dospecials 3778	1043, 1048, 1295, 1296, 1316, 1414
\dot	\familydefault 472,488,504
	\FancyVerbSpace 3824
E	\fi435, 1259, 1262,
\else 433, 1565,	1565, 1566, 1651, 1681, 1720,
1566, 1649, 1679, 1718, 3748, 3776	3538, 3603, 3615, 3621, 3750, 3776
\else: 450	\fi: 452
\em 3703	\file_if_exist:nT 996
\emfontdeclare 3703	\file_if_exist:nTF 377
	\file_input:n 997
\eminnershape	\font 419, 423,
\emph <u>3703</u>	911, 2191, 2208, 2229–2231,
\emreset 3709, 3731, 3736	2237-2239, 2250, 2255, 2260,
\emshape <u>3703</u>	2280, 2290, 2294, 3746, 3749, 3756
\EncodingAccent 3440	\fontdimen 2203, 2229-
\EncodingCommand 3434	2231, 2237–2239, 2250, 2255, 2260
\EncodingComposite 3452	\fontencoding 457, 467, 483, 499, 544
\EncodingCompositeCommand 3458	\fontfamily 468, 484, 500, 543, 604
\encodingdefault 473, 489, 505	\fontname 911,1297
\EncodingSymbol <u>3446</u>	\fontshape 3657, 3680, 3681
\endinput 6, 8, 15	\fontspec 454
environments:	\fontspec_complete_fontname:Nn
listingcont* <u>3806</u>	. 1221, 1224, 2013, 2017, 2027,
verbatim* 3790	2052, 2057, 2061, 2065, 2075, 2141
\etex_iffontchar:D 448	\fontspec_font_if_exist:n 898
\ExecuteOptions 373	\fontspec_font_if_exist:nTF <u>I</u> , 907
\exp_after:wN 1339	\fontspec_if_aat_feature:nn 722
\exp_args:\nx 1596-1598, 1602, 1603	\fontspec_if_aat_feature:nnTF . <u>I</u> , 722
\exp_args:No 996	\fontspec_if_current_feature:n . 908
\exp_args:NV 714	\fontspec_if_current_feature:nTF
\exp_args:Nx 973, 1326, 1624	<u>1</u> ,714,908
\exp_args:Nxx 910	\fontspec_if_current_language:n 867
\exp_not:N 143, 467-469, 483-	\fontspec_if_current_language:nTF
485, 499–501, 541, 543–545,	<u>1</u> , 867
709, 710, 990, 1107, 1436, 1448,	\fontspec_if_current_script:n 850
1461, 2695, 2696, 2699, 2707, 2708	
\exp_not:n 465, 481, 497,	\fontspec_if_current_script:nTF .
693, 969, 1107, 1435, 1447, 1607	<u>1</u> , <u>850</u>
75, 7 % 1, m105, m11/1 ===1	\fontspec_if_feature:n 754
F	\fontspec_if_feature:nnn 782
\f@encoding 3676	\fontspec_if_feature:nnnTF <u>I</u> , <u>782</u>
\f@family 456,	\fontspec_if_feature:nTF <u>I</u> , 754
594, 595, 720, 726, 745, 758, 763,	\fontspec_if_fontspec_font: 718
	· · - /

\fontspec_if_fontspec_font:TF	\g_@@_bfmathrm_tl
<u>1</u> , 590, 718, 724, 743,	64, 517, 3587, 3592–3594
756, 784, 801, 816, 835, 852, 869	\g_@@_cfg_bool 33, 359, 360, 3645
\fontspec_if_language:n 814	\g_@@_curr_series_tl 1534,
\fontspec_if_language:nn 833	2008, 2031, 2034, 2037, 2040, 2087
\fontspec_if_language:nnTF <u>I</u> ,833	\g_@@_default_fontopts_clist
\fontspec_if_language:nTF <u>I</u> , 814	54, 563, 1017
\fontspec_if_opentype: 74I	\g_@@_em_prop
\fontspec_if_opentype:TF <u>I</u> , 74I	62, 3705, 3712, 3713, 3722, 3727
\fontspec_if_script:n 799	\g_@@_euenc_bool
\fontspec_if_script:nTF <u>I</u> , 799	35, 361, 362, 375, 388, 391, 410
\fontspec_maybe_setup_maths: 3599	\g_@@_fontopts_prop 57, 575,
\fontspec_merge_shape:n	578, 582, 583, 994, 1008, 1011, 1314
3669, 3686, 3691, 3696, 3701	\g_@@_hexcol_tl 88, 90, 1130, 1550
\fontspec_new_lang:nn 682, 2769	\g_@@_math_bool
\fontspec_new_script:nn 678, 2732	3615, 3617, 3618, 3621, 3623–3631
\fontspec_parse_colour:viii	\g_@@_math_euler_bool . 30,3515,3540
2314, 2322	\g_@@_math_lucida_bool
\fontspec_parse_cv:w 2695, 2707	
\fontspec_patch_fancyvrb: 3787, 3820	\g_@@_mathrm_tl 63,512,533,
\fontspec_patch_listings: 3788, 3827 \fontspec_patch_moreverb: 3786, 3806	3579, 3580, 3582, 3583, 3586, 3589
\fontspec_patch_werbatim: 3785, 3790	\g_@@_mathsf_tl 65,522,534,3584,3596
\fontspec_print_visible_spaces: .	\g_@@_mathtt_tl 66,527,535,3585,3597
3767, 3782, 3796, 3802, 3816	\g_@@_opacity_tl
\fontspec_select:nn 952	89, 91, 1130, 1549, 2325, 2337
\fontspec_set_em_level:n 3741	\g_@@_OT_features_prop 60, 1578, 1580
\fontspec_set_family:cnn 538	\g_@@_pkg_euler_loaded_bool
\fontspec_set_family:Nnn	32, 3505, 3508, 3514
<u>I</u> , 456, 463,	\g_@@_postadjust_tl 92, 1551 \g_@@_rmfamily_family 463, 464, 468
479, 495, 512, 517, 522, 527, <u>884</u>	\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\fontspec_set_fontface:NNnn \underline{I} , $\overline{891}$	\g_@@_single_feat_tl
\fontspec_setup_maths: 3503, 3634	698, 710, 712, 714, 1509, 1960,
\fontspec_tmp: 412,415	1973, 2743, 2776, 2788, 2798, 2805
\fontspec_visible_space:	\g_@@_ttfamily_family 495,496,500
<u>3754</u> , 3771, 3824, 3831	\g_fontspec_encoding_tl
\fontspec_visible_space_fallback:	393, 394, 396, 400,
3758, <u>3760</u>	401, 403, 404, 407, 408, 1535,
\FontspecSetCheckBoolFalse 24	3579, 3580, 3582–3586, 3589,
\FontspecSetCheckBoolTrue 23	3592-3594, 3596, 3597, 3638, 3763
\fp_eval:n 2195	\Gamma
\fp_new:N	\global 423, 3813 \grave 3525
G	\group_begin:
\g_@@_all_keyval_modules_clist	. 439, 592, 699, 900, 917, 1303,
	1412, 2189, 3534, 3707, 3767, 3796
\g_@@_all_opentype_feature_names_prop	\group_end:
61, 1739–1860	443, 444, 603, 707, 904, 905, 950,
\g_@@_bf_series_seq 53, 2009, 2029, 2032	1307, 1415, 2199, 3539, 3716, 3773

Н	2714, 2723, 2731, 2734, 2768,
\hat 3531	2771, 2790, 3310, 3357, 3369, 3380
\hbox 3776	\keys_if_choice_exist:nnnT . 626,634
hyphenchar 2280, 2290, 2294, 3746, 3749	\keys_if_exist:nnF 624,632
_	\keys_if_exist:nnT 648,666
I	\keys_set:nn 108, 652, 926, 1936,
\IfBooleanTF 562, 573	1946, 1986, 1991, 2360, 2780, 2809
\ifcase 1253	\keys_set:nx 1088,
\IfFontExistsTF 907	1089, 1100, 1101, 1116, 1121, 1126
\IfFontFeatureActiveTF 690	\keys_set_known:nnN 96, 109, 593
\IfFontFeaturesTF <u>690</u>	\keys_set_known:nxN 1339
\ifmmode 3776	
\IfNoValueTF 555	L
\ifnum 1257, 1646, 1676, 1714, 3746	\l_@@_addfontfeatures_bool 2274
\ifx 431,	\l_@@_alias_bool
1565, 1566, 3536, 3603, 3615, 3621	37, 641, 650, 656, 661, 668, 673
\ignorespaces 459, 475, 491, 507, 558, 609	\l_@@_all_features_clist
\ImportEncoding 148	314, 932, 1005, 1015, 1029, 1176
\InputIfFileExists 3647	\l_@@_arg_tl 2156, 2157, 2159, 2164, 2169
\int_case:nnF 2209	\l_@@_atsui_bool 26,
\int_compare:nT 2159, 2344	728, 1120, 1249, 1256, 1265, 1499
\int_compare:nTF	\1_@@_basename_tl 923, 1227
227, 1888, 1956, 2310, 2313	\1_@@_check_bool
\int_compare_p:nNn 1644, 1674, 1712	22–24, 1711, 1716, 1722, 1735
\int_gincr:c	\l_@@_check_feat_bool
\int_if_even:nTF 1893	\1_@@_curr_bfname_tl
\int_incr:N 1650, 1680, 1719, 3714, 3725 \int_new:c	2027, 2035, 2037, 2040, 2087
\int_new:\text{N} \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	\l_@@_curr_fontname_tl 1221,1222
\int set:\n\ \ 42-4/	\l_@@_defining_encoding_bool
. 105, 107, 764, 767, 824, 1539,	40, 3436, 3442, 3448,
1561, 1641, 1648, 1670, 1678,	3454, 3460, 3484, 3486, 3491, 3498
1703, 1717, 2295, 2335, 2741,	\l_@@_disable_defaults_bool
2759, 2775, 2796, 2803, 3723, 3741	36, 596, 1003
\int_set_eq:NN 427	\1_@@_em_int
\int_to_hex:n 2345	46, 3723, 3725, 3727, 3730, 3741
\int_use:c 1157	\1_@@_em_switch_tl 3727,3728
\int_use:N 3713, 3727	\1_@@_em_tmp_tl 3722, 3723
\int_zero:N 1540-1542, 1642,	\1_@@_emdef_int 47,3706,3712-3714
1672, 1710, 2787, 3706, 3730, 3740	\1_@@_ext_filename_tl 992,993,996,997
\itdefault . 1212, 1214, 1423, 1424,	\1_@@_extension_tl
1428, 1440, 1442, 3582, 3589,	962, 981, 1060, 1943, 1949, 1996
3594, 3652, 3661, 3663, 3665, 3686	\l_@@_extensions_clist
\itscdefault 1452, 1453,	
3652, 3657, 3661, 3663, 3665–3667	\1_00_external_bool 38, 1279, 1932, 1944
\itshape 3683,3737	\l_@@_fake_embolden_tl
	69, 2441, 2444, 2458
K	\1_@@_fake_slant_tl 68, 2436, 2463, 2466
\keys_define:nn 614, 651, 669,	\l_@@_family_fontopts_clist
1572, 1587, 1862, 1865, 1921,	1011, 1012, 1018
1951, 2398, 2406, 2415, 2425,	\1_@@_family_label_tl
2430, 2452, 2667, 2680, 2701,	67, 886, 893, 1011, 1013

\	l v = - =
\l_@@_firsttime_bool . 18,704,940,	\1_@@_language_int
1077, 1507, 1527, 2107, 2219,	43, 767, 792, 1183,
2316, 2327, 2339, 2388, 2434, 2456	1707, 1714, 2775, 2787, 2796, 2803
\l_@@_font_path_tl	\l_@@_leftover_clist 1304, 1306
1072, 1073, 1529, 1933	\1_@@_mm_bool 28,1251,1258,2381,2386
\l_@@_fontdef_tl 726,727,745,746,	\l_@@_never_check_bool
758, 759, 786, 787, 803, 804, 818,	39, 703, 1636, 1665, 1698
819, 837, 838, 854, 855, 871, 872	\1_@@_nfss_enc_tl 457,
\l_@@_fontfeat_bf_clist	467, 473, 483, 489, 499, 505, 544,
	1168, 1391, 1398, 1427, 1535, 2145
\l_@@_fontfeat_bfit_clist 80,	\1_@0_nfss_fam_tl 1141, 1143, 2149
1214, 2095, 2443, 2445, 2465, 2467	\1_@@_nfss_prop 58, 2039, 2086
\l_@@_fontfeat_bfsl_clist	\l_@@_nfss_sc_tl
82, 1215, 2103	1323, 1371, 1396, 1399, 1458
\1_00_fontfeat_clist 705, 1036, 1109	\l_@@_nfss_tl 1322,1346,1392,1446
\l_@@_fontfeat_curr_clist	\l_@@_nfssfont_prop 59,1217,1240
	\1_@@_nobf_bool 19,
\l_@@_fontfeat_it_clist	1189, 1196, 1930, 2023, 2026, 2461
79, 1212, 2091, 2437	\l_@@_noit_bool 20,
\l_@@_fontfeat_sc_clist 83, 1350, 2109	1189, 1201, 1931, 2048, 2051, 2439
\l_@@_fontfeat_sl_clist 81,1213,2099	\1_@@_nosc_bool
\l_@@_fontfeat_up_clist	21, 1352, 1363, 1369, 2071, 2074
77, 1210, 2080, 2115	\1_00_opacity_tl 1129,
\l_@@_fontid_tl 932, 934,	1131, 1549, 2325, 2330, 2337, 2342
1139, 1143, 1145, 1155, 1160, 2150	\l_@@_optical_size_tl
\l_@@_fontname_bf_tl 71,	1063, 1530, 2378, 2395
984, 1192, 1198, 1211, 2035, 2460	\1_@@_options_tl 594,597,601
\l_@@_fontname_bfit_tl 73,985,	\l_@@_ot_bool 27,
1191–1193, 1214, 2057, 2446, 2468	702, 748, 761, 789, 806, 821, 840,
\l_@@_fontname_bfsl_tl	857, 874, 1113, 1250, 1261, 1268,
75, 1206, 1215, 2065	1274, 1468, 1496, 1526, 2376, 2386
\l_@@_fontname_it_tl 72,	\1_@@_postadjust_tl
983, 1191, 1203, 1212, 2052, 2438	1392, 1399, 1428,
\1_@@_fontname_sc_tl	1459, 1461, 1551, 2266, 2279, 2288
76, 1354, 1366, 2075	\1_00_pre_feat_sclist 1180, 1386, 1465
\l_@@_fontname_sl_tl	\1_@@_prev_unicode_name_tl 3482, 3487
74, 1206, 1213, 2061	\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\l_@@_fontname_tl 595, 597, 601	\1_@@_proceed_bool 1609, 1618, 1622
\\1_@@_fontname_up_tl 393, 397, 001	
	\l_@@_punctspace_adjust_tl
70, 922, 1032, 1041, 1043, 1044, 1046, 1048, 2013, 2017	92, 1553, 2249, 2254, 2259
	\l_@@_rawfeatures_sclist
\l_@@_fontopts_clist	706, 709, 939, 1133, 1180,
1008, 1009, 1019, 1306, 1314, 1315	1379, 1380, 1386, 1511, 1520, 1547
\1_@0_graphite_bool 29, 1252	\1_@@_saved_fontname_tl 1324, 1337
\l_@@_hexcol_tl	\1_@@_scale_tl
1129, 1131, 1550, 2307, 2311, 2324	. 325, 1384, 1548, 2184, 2185, 2193
\l_@@_keys_leftover_clist	\l_@@_script_exist_bool
. 1030, 1033–1035, 1110, 1111,	
1115, 1116, 1119, 1121, 1125, 1126	\l_@@_script_int
\l_@@_lang_name_tl . 86, 263, 1084,	42, 764, 791, 824, 843, 1182,
1086, 1089, 1096, 1098, 1101, 1992	1671, 1676, 1706, 1714, 2741, 2759

	I
\l_@@_script_name_tl 84, 263,	\l_fontspec_fontname_tl
273, 1079, 1083, 1088, 1100, 1987	173, 237, 242,
\l_@@_size_tl	247, 252, 257, 262, 267, 272, 325,
1336, 1341, 1342, 1377, 1384, 2137	333, 921, 929, 932, 1008, 1032,
\l_@@_sizedfont_tl	1179, 1193, 1198, 1203, 1210,
1337, 1345, 1347, 2141	1313, 1314, 1316, 1321, 1324,
\l_@@_sizefeat_clist	1377, 1385, 2438, 2446, 2460, 2468
56, 1235, 1241, 2114, 2119	\l_fontspec_hyphenchar_tl
\l_@@_sizing_leftover_clist	2284–2286, 2290, 2295
	\l_fontspec_lang_tl
\l_@@_strnum_int 44,	87, 770, 1185, 1473,
1640, 1646, 1669, 1676, 1709,	1484, 1732, 2774, 2786, 2797, 2804
1715, 2741, 2759, 2775, 2796, 2803	\l_fontspec_mode_tl
\l_@@_tfm_bool 25, 1248, 1254	1480, 1538, 1971, 1973
\l_@@_this_feat_tl 2157, 2170, 2175	\l_fontspec_renderer_tl 1062,
	1263, 1266, 1269, 1532, 1958, 1960
\l_@@_this_font_tl 1232,1238,1241, 2120, 2121, 2125, 2158, 2169, 2175	\l_fontspec_script_tl 85,
	769, 825, 842, 1184, 1470, 1472,
\1_@@_tmp_int 45, 2335, 2336, 2344, 2345	1481, 1483, 1690, 1732, 2740, 2758
\1_@@_tmp_tl 570, 571, 575,	\l_keys_choice_int 1956
578, 582, 583, 593, 763, 764, 766,	\l_keys_choice_tl 1959, 1972
767, 823, 824, 859, 860, 876, 877,	
1148, 1149, 1151–1153, 1157, 1236	\1_keys_key_t1 246, 251, 256, 261
\1_00_tmpa_bool 972,975,977	\l_keys_value_tl 246, 251, 256, 261
\l_@@_tmpa_dim 50,2191,2195	\1_tmpa_font 1295, 1297
\1_@@_tmpa_fp 48	\1_tmpa_int 1642, 1644, 1646, 1648,
\1_@@_tmpb_dim 51, 2192, 2196	1650, 1672, 1674, 1676, 1678,
\l_@@_tmpb_fp 49	1680, 1710, 1712, 1715, 1717, 1719
\1_@@_tmpb_tl 575-578	\1_tmpa_tl 1884, 1885, 1908
\l_@@_tmpc_dim 52	\1_tmpb_font 1296, 1297
\l_@@_ttc_index_tl	\1_tmpb_int 1641, 1644, 1648,
1061, 1531, 1998, 1999, 2003, 2004	1670, 1674, 1678, 1703, 1712, 1717
\l_@@_wordspace_adjust_tl	\1_tmpb_t1
92, 1552, 2227, 2235	1890, 1895, 1898, 1902, 1905, 1908
\l_fontspec_check_bool 1643, 1647,	\Lambda 3561
1653, 1659, 1673, 1677, 1683, 1692	\latinencoding 404, 408
\lfontspec_em_int 3740	\leavevmode 3776
\l_fontspec_defined_shapes_tl	\let 3778
317, 1433, 1533	\liningnums <u>3834</u>
\l_fontspec_family_tl 313,	\listing@line 3813
604, 888, 896, 1160, 1161, 1168,	listingcont*(environment) 3806
1174-1177, 1182-1185, 1391,	\lst@visiblespace 3831
1398, 1427, 1428, 2151, 2152, 3642	\luatex_postexhyphenchar:D 1542
\l_fontspec_feature_string_tl	\luatex_posthyphenchar:D 1540
	\luatex_preexhyphenchar:D 1541
\l_fontspec_font	\luatex_prehyphenchar:D 1539, 2295
727, 746, 759, 787, 804,	
819, 838, 855, 872, 895, 1042,	M
1044, 1047, 1049, 1253, 1257,	\mathalpha 3524-3533, 3548-3568
1316, 1317, 1641, 1646, 1671,	\mathbf 183, 3583, 3593
10/0, 1/05, 1/14, 1004, 1000,	
1676, 1705, 1714, 1884, 1888, 1890, 1895, 1900, 2192, 2286, 3643	

	I
\mathclose 3542, 3545, 3572, 3574	\numexpr 1900
\mathdollar 3576	
\mathit 183, 3582, 3589, 3594, 3685	O
\mathopen 3571, 3573	\oldstylenums <u>3834</u>
\mathord 3575, 3576	\Omega 3568
\mathpunct 3537, 3544	\or 1255, 1260
\mathrel 3543, 3570	
\mathring 3533	P
\mathrm 3581, 3592	\par 3814
\mathsf 3584, 3596	\Path 1928
\mathtt 3585, 3597	\Phi 3566
00 0 001	\Pi 3563
\mddefault	\prg_new_conditional:Nnn 446,
. 1210, 1212, 1213, 1438, 1440,	696, 718, 722, 741, 754, 782, 799,
1441, 1450, 1452, 1454, 3579,	
3580, 3582, 3584, 3585, 3592, 3594	814, 833, 850, 867, 898, 908, 967,
\msg_error:nn 124	1135, 1293, 1634, 1663, 1696, 1882
\msg_error:nnn 125	\prg_return_false:
\msg_error:nnx 126	434, 443, 451, 712, 715,
\msg_fatal:nn 14	720, 731, 734, 738, 748, 751, 772,
\msg_info:nn 130	775, 779, 793, 795, 797, 808, 810,
\msg_info:nnx 131	812, 827, 829, 831, 844, 846, 848,
\msg_info:nnxx 132	861, 863, 865, 878, 880, 882, 905,
\msg_line_context: 223	912, 977, 1162, 1298, 1653, 1659,
\msg_new:nnn 9, 134	1683, 1692, 1722, 1735, 1886, 1906
\msg_new:nnnn 135	\prg_return_true:
\msg_new:nnx 137	432, 444, 449, 715, 720,
\msg_new:nnxx	731, 748, 772, 793, 808, 827, 844,
\msg_redirect_module:nnn	861, 878, 904, 912, 977, 1162,
365, 366, 370, 371	1299, 1637, 1653, 1659, 1666,
\msg_redirect_name:nnn 2359	1683, 1692, 1699, 1722, 1735, 1909
	\prg_set_conditional:\Nnn \ldots 429,437
	\ProcessOptions 374
	\prop_clear:N 3705
\msg_warning:nnx 128	\prop_get:cnN 594, 595, 726, 745, 758,
\msg_warning:nnxx 129	763, 766, 769, 770, 786, 803, 818,
N	823, 825, 837, 854, 859, 871, 876
	\prop_get:NnNT 114
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	\prop_get:NnNTF 115
NewDocumentCommand 454, 3838	\prop_get:NVNF 575, 1008, 1011, 1314
\newfontface	\prop_get:NxNT
\newfontfamily <u>536</u> , 551	1
\newfontfeature $\underline{612}$	\prop_get:NxNTF 3727 \prop_gput:cnV 1182-1185
\newfontlanguage <u>680</u> , 2907-3288	1 1 - 01
\newfontscript 676, 2835-2906	\prop_gput:cnx 1175-1177
\newICUfeature 638	\prop_gput:Nnn
\newopentypefeature 630, 638	112, 113, 1580, 1739–1860
	\prop_gput:NVV 578
\normalfont . 474, 490, 506, 3708, 3740	\prop_gput:Nxn 3713
\normalsize 402, I4I3	\prop_gput:NxV 3712
\not@math@alphabet	\prop_gremove:NV 582
3656, 3685, 3690, 3695, 3700	\prop_if_in:NnF 1578
\null 3776	\prop_if_in:NVF 994
\nullfont 431	\prop_map_inline:Nn 1217

	I
\prop_new:c 1174	\sldefault
\prop_new:N 57-62	. 1213, 1215, 1424, 1427, 1441,
\prop_put:Nnn 110, 111	1443, 3653, 3662, 3664, 3666, 3691
\prop_put:NVn 583	\slscdefault 1454,
\prop_put:NxV 2039, 2086	1455, 3653, 3662, 3664–3666, 3668
\prop_put:Nxx 1240	\slshape <u>3683</u>
\providecommand 3426-3431, 3652, 3653	\space 325, 1890, 1895, 1900
\Psi 3567	\str_case:nn 1436, 1448, 1633
0	\str_case:nnF 2179
Q	\str_case_x:nnF 2245
\q_nil 1557, 1559, 2696, 2708	\str_if_eq:nnTF
\q_stop 100, 101, 2220, 2223	116, 2208, 2277, 2356, 2750
R	\str_if_eq:nVTF 860,877
\relax 1900, 3430,	\str_if_eq_x:nnF 1054, 1129, 1996
3535, 3656, 3690, 3695, 3700, 3776	\str_if_eq_x:nnT 472, 488, 504, 981
\RenewDocumentCommand 3834	\str_if_eq_x:nnTF 1297
\RequirePackage 4, 400, 414	\str_if_eq_x_p:nn 1423, 1424
\RequirePackageWithOptions 6,8	\str_lower_case:f 981,1996
\reset@font 3740	\string 144, 148, 183, 203, 223
\rmdefault 397, 464, 472, 533	\sys_if_engine_luatex:T 3
\rmfamily 465, 2190, 2208	\sys_if_engine_xetex:T 7
1 3/ 7 /	(byb_ii_engine_ketekii/
S	T
\scan_stop: . 419, 423, 448, 2290, 3757	\textsi 3652
\scdefault	\textvisiblespace 3764
. 1405–1407, 1450, 1451, 3652,	\the
3653, 3661–3664, 3667, 3668, 3696	
\scshape <u>3683</u>	\thelisting@line 3813 \Theta 3560
\selectfont 458, 469,	
485, 501, 545, 604, 3657, 3680, 3681	\tilde 3527
\seq_gput_right:Nx 2009	\tl_clear:N 576, 1013, 1322, 1323, 1336, 1529-1534, 1547,
\seq_if_empty:NT 2029	1548, 1552, 1553, 1949, 2158, 2170
\seq_new:N 53	\tl_const:cn
\seq_put_right:Nx 2032	
\setboldmathrm <u>510</u>	\tl_count:n 2310, 2313
	\tl_count:n 2310, 2313 \tl_gclear:N 698
\setboldmathrm <u>510</u>	\tl_count:n
\setboldmathrm	\tl_count:n 2310, 2313 \tl_gclear:N 698 \tl_gput_right:Nx 1433, 1511 \tl_gremove_all:Nn 1520
$\begin{tabular}{lllllllllllllllllllllllllllllllllll$	\tl_count:n 2310, 2313 \tl_gclear:N 698 \tl_gput_right:Nx 1433, 1511 \tl_gremove_all:Nn 1520 \tl_gset:cx 1155
\setboldmathrm	\tl_count:n 2310, 2313 \tl_gclear:N 698 \tl_gput_right:Nx 1433, 1511 \tl_gremove_all:Nn 1520 \tl_gset:cx 1155 \tl_gset:Nn 2788, 2798, 2805
\setboldmathrm 510 \setmainfont 461, 509 \SetMathAlphabet 3582- 3585, 3589, 3592-3594, 3596, 3597 \setmathrm 510 \setmathsf 510	\tl_count:n 2310, 2313 \tl_gclear:N 698 \tl_gput_right:Nx 1433, 1511 \tl_gremove_all:Nn 1520 \tl_gset:cx 1155 \tl_gset:Nn 2788, 2798, 2805 \tl_gset:Nv 1160
\setboldmathrm \ \frac{510}{510} \setmainfont \ \frac{461}{509} \SetMathAlphabet \ \ 3582- \ \ 3585, 3589, 3592-3594, 3596, 3597 \setmathrm \ \ \frac{510}{510} \setmathtf \ \ \frac{510}{510} \setmathtf \ \ \frac{510}{510} \setmathff \ \ \frac{510}{510} \setmathfff \ \ \frac{510}{510} \setmathffff \ \ \frac{510}{510} \setmathfffff \ \ \frac{510}{510} \setmathfffff \ \ \frac{510}{510} \setmathfffff \ \ \frac{510}{510} \setmathffffff \ \ \frac{510}{510} \setmathfffffffffffffffffffffffffffffffffff	\tl_count:n
\setboldmathrm 510 \setmainfont 461,509 \SetMathAlphabet 3582- 3585,3589,3592-3594,3596,3597 \setmathrm 510 \setmathsf 510 \setmathtt 510 \setmonofont 493	\tl_count:n
\setboldmathrm 510 \setmainfont 461, 509 \SetMathAlphabet 3582- 3585, 3589, 3592-3594, 3596, 3597 \setmathrm 510 \setmathsf 510 \setmathtt 510 \setmathtt 510 \setmonofont 493 \setromanfont 509	\tl_count:n
\setboldmathrm 510 \setmainfont 461, 509 \SetMathAlphabet 3582- 3585, 3589, 3592-3594, 3596, 3597 \setmathrm 510 \setmathsf 510 \setmathtt 510 \setmonofont 493 \setromanfont 509 \setsansfont 477	\tl_count:n
\setboldmathrm 510 \setmainfont 461, 509 \SetMathAlphabet 3582- 3585, 3589, 3592-3594, 3596, 3597 \setmathrm 510 \setmathsf 510 \setmathtt 510 \setmonofont 493 \setromanfont 509 \setsansfont 477 \SetSymbolFont 3523, 3580, 3586	\tl_count:n
\setboldmathrm 510 \setmainfont 461, 509 \SetMathAlphabet 3582- 3585, 3589, 3592-3594, 3596, 3597 \setmathrm 510 \setmathsf 510 \setmathtt 510 \setmonofont 493 \setromanfont 509 \setsansfont 477 \SetSymbolFont 3523, 3580, 3586 \settoheight 2206	\tl_count:n
\setboldmathrm \ \ \frac{510}{509} \setmainfont \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	\tl_count:n
\setboldmathrm \ \ \frac{510}{509} \setmainfont \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	\tl_count:n
\setboldmathrm \ \ \frac{510}{509} \setmainfont \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	\tl_count:n

\tl_if_empty:NTF	\token_to_str:N 2305, 3501
. 712, 1079, 1354, 1885, 1905, 3587	\ttdefault 399, 496, 504, 535
\tl_if_empty:nTF 117,	\ttfamily 497
581, 1872, 2021, 2046, 2069, 2225	\two@digits 2687, 2699
\tl_if_empty:xF 1281	\typeout95,97,
\tl_if_empty:xTF 1234	589, 597, 692, 693, 709, 710, 916,
\tl_if_eq:NNF 2325, 2337	934, 948, 969, 990, 1002, 1026,
\tl_if_eq:nnT 120	1040, 1041, 1046, 1107, 1115,
\tl_if_eq:oxT 2034	1119, 1125, 1138, 1139, 1246,
\tl_if_exist:cTF 1405	1312, 1321, 1342, 1347, 1358,
\tl_if_exist:NT 2151	1362, 1377, 1380, 1411, 1506,
\tl_if_exist_p:c 3673	1510, 1516, 1519, 1525, 1607,
\tl_if_in:NnF 2164	1702, 1727, 2037, 2162, 2167, 3649
\tl_if_in:NnT 402, 959	U
\tl_if_in:nnT 974	l .
\tl_if_in:nnTF 910	<u></u>
\tl_if_single:nTF 569, 2283	\UndeclareSymbol 3489
\tl_new:N 63-89	\UndeclareTextCommand 3493
\tl_put_left:Nn 1902 \tl_put_right:Nn 577, 2279, 2288	\unexpanded 1002
\tl_put_right:Nx 1382, 2266	\UnicodeEncodingName
\tl_remove_all:Nn	3438, 3444, 3450, 3456, 3462, 3482, 3483, 3487, 3493, 3501
687, 956, 993, 1149, 1228	\UnicodeFontFile
\tl_remove_once:Nn 961	3426, 3472, 3474, 3476, 3478, 3480
\tl_replace_all:Nnn 121	\UnicodeFontName 3427
\tl_replace_all:Nnx 1227	\UnicodeFontTeXLigatures 3428,
\tl_set:cn 1978-1983	3429, 3472, 3474, 3476, 3478, 3480
\tl_set:Nn 90-92, 393, 394, 396-	\updefault . 1210, 1211, 1438, 1439,
399, 533–535, 842, 886, 893, 962,	3579, 3580, 3583–3586, 3592,
1083, 1086, 1098, 1266, 1269,	3593, 3596, 3597, 3667, 3668, 3701
1538, 1933, 1943, 1987, 1992,	\upshape <u>3683</u> , 3736, 3738
1998, 1999, 2003, 2004, 2121,	\Upsilon 3565
2125, 2137, 2227, 2235, 2249,	\use:c 226,543
2254, 2259, 2284, 2285, 2311,	\use:n 1068, 1339
2324, 2330, 2378, 2395, 2436,	\use:x 465, 481, 497, 539, 598, 2693
2458, 2740, 2758, 2774, 2786, 2797, 2804, 3483, 3638, 3642, 3643	\use_i:nnn 2175
\tl_set:No 570, 2156	\use_ii:nnn 2175
\tl_set:Nv	\use_iii:nnn 2156
\tl_set:Nx 932, 955, 992, 1148, 1226,	\usefont 3763
1232, 1286, 1884, 1890, 1895,	\UTFencname 401
1898, 1908, 2149, 2184, 2185, 2342	V
\tl_set_eq:NN 401, 403, 404, 407, 408,	
464, 473, 480, 489, 496, 505, 888,	<u> </u>
895, 896, 1032, 1133, 1324, 1337,	\verb* 3774
1549-1551, 2035, 2157, 2169,	\verb@eol@error 3778
2438, 2446, 2460, 2468, 3482, 3487	verbatim* (environment) 3790
\tl_to_str:N 932	\verbatim@font 3779
\tl_to_str:n 911, 3501	\verbatim@line 3814
\tl_trim_spaces:n 137, 139	\verbatim@processline 3811
\tl_use:c 1406, 3677, 3680	\verbatim@start 3796, 3816

X	\XeTeXOTscripttag 1646
\xetex_suppressfontnotfounderror:D	\XeTeXpicfile 412,413,415
427	\XeTeXselectorname 1890, 1895, 1900
\XeTeXcountvariations 1257	\Xi 3562
\XeTeXfeaturename 1884	\xlx@defaulthyphenchar 3747,3753
\XeTeXfonttype 1253	
\XeTeXisexclusivefeature 1888	Z
\XeTeXOTcountfeatures 1705	\z@ 3746
\XeTeXOTcountlanguages 1671	\zf@basefont
\XeTeXOTcountscripts 1641	\zf@enc 3638
\XeTeXOTfeaturetag 1714	\zf@family 3638
\XeTeXOTlanguagetag 1676	\zf@fontspec 3638