The XeTeX reference guide

Will Robertson Khaled Hosny

April 27, 2014

Introduction

This document serves to summarise additional features of XeTeX without being so much as a 'users' guide. Note that much of the functionality addressed here is provided in abstracted form in various LaTeX packages and ConTeXt modules.

The descriptions here should be a fairly exhaustive list of the new primitives and features of XeTeX. Descriptions are still a little aenemic, however.

License

Contents

Part I
XeTeX specifics

1 The `\font` command

Traditionally, fonts were selected in TeX like this: `\font\1=`⟨*tfm name*⟩ with various options possibly appended such as ‘`at 10pt`’ or ‘`scaled 1.2`’, with obvious meaning. This syntax still works, but it has been greatly extended in XeTeX.

The extended syntax looks schematically like this:

`\font\1="`⟨*font identifier*⟩⟨*font options*⟩`:`⟨*font features*⟩`"` ⟨*TeX font options*⟩

The ⟨*font identifier*⟩ is the only mandatory part of the above syntax. If it is given in square brackets, (e.g., `[lmroman10-regular]`), it is taken as a font file name. Without brackets, the name is looked up both as a file name and a system font name. When using a font name, the font is looked up through the operating system, using (except on Mac OS X) the `fontconfig` library. Running `fc-list` should show you the font names available. *E.g.*,

```
\font\1="Liberation Serif"                    look for OS-installed font
```

Fonts have many internal names, and XeTeX matches them in the following order:

- Full Name;
- if the name has a hyphen, it is split into Family-Style pair then matched;
- PostScript Name;
- Family Name, if there is more than one match;
  - look for font with “regular” bit set in OS/2 table, if no match;
  - look for font with style “Regular”, “Plain”, “Normal” or “Roman”, in that order.

When using a file name, the `xdvipdfmx` driver must be used (this is the default). The current directory and the `texmf` trees are searched for files matching the name, or the path may be embedded in the font declaration, as usual with `kpathsea`. *E.g.*,

```
\font\2="[lmroman10-regular]"          find lmroman10-regular.otf in any tree
\font\3="[/myfonts/fp9r8a]"                look for fp9r8a only in /myfonts/
```

A file with either an `.otf`, `.ttf` or `.pfb` extension (in that order) will be found. The extension can also be specified explicitly. If the file is a font collection (e.g., `.ttc` or `.dfont`), the index of the font can be specified using a colon followed by zero-based font index inside the square brackets. *E.g.*,

```
\font\4="[myfont.ttc:1]"                load the second font from myfont.ttc file
```

1.1 *Font options*

⟨*Font options*⟩ are only applicable when the font is selected through the operating system (*i.e.*, without square brackets). They may be any concatenation of the following:

/B	Use the bold version of the selected font.
/I	Use the italic version of the selected font.
/BI	Use the bold italic version of the selected font.
/IB	Same as `/BI`.
/S=*x*	Use the version of the selected font corresponding to the optical size *x* pt.
/AAT	Explicitly use the AAT renderer (Mac OS X only).
/OT	Explicitly use the OpenType renderer (new in 0.9999).
/GR	Explicitly use the Graphite renderer.[1]
/ICU	Explicitly use the OpenType renderer (deprecated since 0.9999).

1.2 *Font features*

The ⟨*font features*⟩ is a comma or semi-colon separated list activating or deactivating various OpenType, Graphite, or AAT font features, which will vary by font. In contrast to font options, features work whether the font is selected by file name or through the operating system.

The XeTeX documentation files `aat-info.tex` and `opentype-info.tex` provide per-font lists of supported features.

1.2.1 *Arbitrary OpenType, Graphite, or AAT features*

OpenType font features are chosen with standard tags[2]. They may be either comma- or semicolon-separated, and prefixed with a + to turn them on and a - to turn them off, optionally followed by = and a 0-based index for selecting alternates from multiple alternates features (ignored for - prefixed tags).

> *Example:*
>
> ```
> \font\liber="Linux Libertine O/I=5:+smcp" at 12pt
> \liber This is the OpenType font Linux Libertine in italic with small caps.
> ```
>
> THIS IS THE OPENTYPE FONT LINUX LIBERTINE IN ITALIC WITH SMALL CAPS.

Varying depending on the language and script in use (see section §1.2.3 on page 6), a small number of OpenType features, if they exist, will be activated by default.

[1] `http://scripts.sil.org/cms/scripts/page.php?site_id=projects&item_id=graphite_home`
[2] `http://www.microsoft.com/typography/otspec/featuretags.htm`

Example:

```
\font\antt="Antykwa Torunska"          at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=0" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=1" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=2" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=3" at 12pt \antt 0
\font\antt="Antykwa Torunska:+aalt=4" at 12pt \antt 0
```

0 o 0 0 0 0

AAT font features and Graphite font features are specified by strings within each font rather than standardised tags. Therefore, even equivalent features between different fonts can have different names.

Example:

```
\font\gra="Charis SIL/GR:Small Caps=True" at 12pt
\gra This is the Graphite font Charis SIL with small caps.
```

THIS IS THE GRAPHITE FONT CHARIS SIL WITH SMALL CAPS.

1.2.2 Options for all fonts

Some font features may be applied for any font. These are

`mapping=<font map>`
: Uses the specified font mapping for this font. This uses the TECKit engine to transform unicode characters in the last-minute processing stage of the source. For example, `mapping=tex-text` will enable the classical mappings from ugly ascii ``` ``---'' ``` to proper typographical glyphs "—", and so on.

`color=RRGGBB[TT]`
: Triple pair of hex values to specify the colour in RGB space, with an optional value for the transparency.

`letterspace=`*x*
: Adds x/S space between letters in words, where S is the font size.

`embolden=`*x*
: Increase the envelope of each glyph by the set amount (this makes the letters look 'more bold'). $x = 0$ corresponds to no change; $x = 1.5$ is a good default value.

`extend=`*x*
: Stretch each glyph horizontally by a factor of x (i.e., $x = 1$ corresponds to no change).

`slant=`*x*
: Slant each glyph by the set amount. $x = 0$ corresponds to no change; $x =$

0.2 is a good default value. The slant is given by $x = R/S$ where R is the displacement of the top edge of each glyph and S is the point size.

1.2.3 OpenType script and language support

OpenType font features (and font behaviour) can vary by script[3] ('alphabet') and by language[4]. These are selected with four and three letter tags, respectively.

`script=<script tag>`
: Selects the font script.

`language=<lang tag>`
: Selects the font language.

1.2.4 Multiple Master and Variable Axes AAT font support

`weight=`*x*
: Selects the normalised font weight, x.

`width=`*x*
: Selects the normalised font width, x.

`optical size=`*x*
: Selects the optical size, x pt. Note the difference between the `/S` font option, which selects discrete fonts.

1.2.5 Vertical typesetting

`vertical`
: Enables glyph rotation in the output so vertical typesetting can be performed.

Part II
New commands

2 Font primitives

`\XeTeXtracingfonts`
If nonzero, reports where fonts are found in the log file.

[3] `http://www.microsoft.com/typography/otspec/scripttags.htm`
[4] `http://www.microsoft.com/typography/otspec/languagetags.htm`

`\XeTeXfonttype` ⟨*font*⟩
Expands to a number corresponding to which renderer is used for a ⟨*font*⟩:

0 for TEX (a legacy TFM-based font);
1 for AAT;
2 for OpenType;
3 for Graphite.

> *Example:*
>
> ```
> \newcommand\whattype[1]{%
>   \texttt{\fontname#1} is rendered by
>   \ifcase\XeTeXfonttype#1\TeX\or AAT\or OpenType\or Graphite\fi.\par}
> \font\1="cmr10"
> \font\2="Charis SIL"
> \font\3="Charis SIL/OT"
> \whattype\1 \whattype\2 \whattype\3
> ```
>
> cmr10 is rendered by TEX.
> "Charis SIL" is rendered by OpenType.
> "Charis SIL/OT" is rendered by OpenType.

`\XeTeXfirstfontchar` ⟨*font*⟩
Expands to the code of the first character in ⟨*font*⟩.

`\XeTeXlastfontchar` ⟨*font*⟩
Expands to the code of the last character in ⟨*font*⟩.

> *Example:*
>
> ```
> \font\1="Charis SIL"\1
> The first character in Charis SIL is: "\char\XeTeXfirstfontchar\1"
> and the last character is: "\char\XeTeXlastfontchar\1".
> ```
>
> The first character in Charis SIL is: " " and the last character is: "𝔓".

`\XeTeXglyph` ⟨*glyph slot*⟩
Inserts the glyph in ⟨*glyph slot*⟩ of the current font. **Font specific**, so will give different output for different fonts and possibly even different versions of the same font.

`\XeTeXcountglyphs` ⟨*font*⟩
The count of the number of glyphs in the specified ⟨*font*⟩.

`\XeTeXglyphname` ⟨*font*⟩ ⟨*glyph slot*⟩
Expands to the name of the glyph in ⟨*glyph slot*⟩ of ⟨*font*⟩. **Font specific**, so will give different output for different fonts and possibly even different versions of the same font.

`\XeTeXglyphindex "`⟨*glyph name*⟩`"` ⟨*space*⟩ *or* `\relax`
Expands to the glyph slot corresponding to the (possibly font specific) ⟨*glyph name*⟩ in the currently selected font. Only works for TrueType fonts (or TrueType-based OpenType fonts) at present. Use `fontforge` or similar to discover glyph names.

`\XeTeXcharglyph` ⟨*char code*⟩
Expands to the default glyph number of character ⟨*char code*⟩ in the current font, or 0 if the character is not available in the font.

> *Example:*
>
> ```
> \font\1="Charis SIL"\1
> The glyph slot in Charis SIL for the Yen symbol is:
>     \the\XeTeXglyphindex"yen" . % the font-specific glyph name
> Or: \the\XeTeXcharglyph"00A5.   % the unicode character slot
>
> This glyph may be typeset with the font-specific glyph slot:
> \XeTeXglyph150,
> or the unicode character slot:
> \char"00A5.
> ```
>
> The glyph slot in Charis SIL for the Yen symbol is: 150. Or: 150.
> This glyph may be typeset with the font-specific glyph slot: ¥, or the unicode character slot: ¥.

`\XeTeXglyphbounds` ⟨*edge*⟩ ⟨*glyph slot*⟩
Expands to a dimension corresponding to one of the bounds of a glyph, where ⟨*edge*⟩ is an integer from 1 to 4 indicating the left/top/right/bottom edge respectively, and ⟨*glyph slot*⟩ is an integer glyph index in the current font (only valid for non TFM-based fonts).

The left and right measurements are the glyph sidebearings, measured 'inwards' from the origin and advance respectively, so for a glyph that fits completely within its 'cell' they will both be positive; for a glyph that 'overhangs' to the left or right, they will be negative. The actual width of the glyph's bounding box, therefore, is the character width (advance) minus both these sidebearings.

The top and bottom measurements are measured from the baseline, like TEX 's height and depth; the height of the bounding box is the sum of these two dimensions.

Example:

```
\def\shadebbox#1{%
\leavevmode\rlap{%
  \dimen0=\fontcharwd\font`#1%
  \edef\gid{\the\XeTeXcharglyph`#1}%
  \advance\dimen0 by -\XeTeXglyphbounds1 \gid
  \advance\dimen0 by -\XeTeXglyphbounds3 \gid
  \kern\XeTeXglyphbounds1 \gid
  \special{color push rgb 1 1 0.66667}%
  \vrule width \dimen0
         height \XeTeXglyphbounds2 \gid
         depth \XeTeXglyphbounds4 \gid
  \special{color pop}%
  \kern\XeTeXglyphbounds3 \gid}%
  #1}

\noindent
\font\x="Charis SIL/I" at 24pt \x
\shadebbox{A} \shadebbox{W} \shadebbox{a} \shadebbox{f}
\shadebbox{;} \shadebbox{*} \shadebbox{=}
```

*A W a f ; * =*

`\XeTeXuseglyphmetrics`
Counter to specify if the height and depth of characters are taken into account while typesetting (≥ 1). Otherwise (< 1), a single height and depth for the entire alphabet is used. Gives better output but is slower. Activated (≥ 1) by default.

Example:

```
\XeTeXuseglyphmetrics=0 \fbox{a}\fbox{A}\fbox{j}\fbox{J} vs.
\XeTeXuseglyphmetrics=1 \fbox{a}\fbox{A}\fbox{j}\fbox{J}
```

a A j J vs. a A j J

2.1 *OpenType fonts*

`\XeTeXOTcountscripts` ⟨*font*⟩
Expands to the number of scripts in the ⟨*font*⟩.

`\XeTeXOTscripttag` ⟨*font*⟩ ⟨*integer, n*⟩
Expands to the *n*-th script tag of ⟨*font*⟩.

`\XeTeXOTcountlanguages` ⟨*font*⟩ ⟨*script tag*⟩
Expands to the number of languages in the script of ⟨*font*⟩.

`\XeTeXOTlanguagetag` ⟨*font*⟩ ⟨*script tag*⟩ ⟨*integer, n*⟩
Expands to the *n*-th language tag in the script of ⟨*font*⟩.

`\XeTeXOTcountfeatures` ⟨*font*⟩ ⟨*script tag*⟩ ⟨*language tag*⟩
Expands to the number of features in the language of a script of ⟨*font*⟩.

`\XeTeXOTfeaturetag` ⟨*font*⟩ ⟨*script tag*⟩ ⟨*language tag*⟩ ⟨*integer, n*⟩
Expands to the *n*-th feature tag in the language of a script of ⟨*font*⟩.

2.2 AAT and Graphite fonts

2.2.1 Features

`\XeTeXcountfeatures` ⟨*font*⟩
Expands to the number of features in the ⟨*font*⟩.

`\XeTeXfeaturecode` ⟨*font*⟩ ⟨*integer, n*⟩
Expands to the feature code for the *n*-th feature in the ⟨*font*⟩.

`\XeTeXfeaturename` ⟨*font*⟩ ⟨*feature code*⟩
Expands to the name corresponding to the ⟨*feature code*⟩ in the ⟨*font*⟩.

`\XeTeXisexclusivefeature` ⟨*font*⟩ ⟨*feature code*⟩
Expands to a number greater than zero if the feature of a font is exclusive (can only take a single selector).

`\XeTeXfindfeaturebyname` ⟨*font*⟩ ⟨*feature name*⟩
This command provides a method to query whether a feature name corresponds to a feature contained in the font. It represents an integer corresponding to the feature number used to access the feature numerically. If the feature does not exist, the integer is `-1`. Also see `\XeTeXfindselectorbyname`.

> *Example:*
>
> ```
> \font\1="Charis SIL/GR" at 10pt
> \def\featname{Uppercase Eng alternates}
> The feature ‘\featname’ has index
> \the\XeTeXfindfeaturebyname\1 "\featname"\relax
> ```
>
> The feature ‘Uppercase Eng alternates’ has index 1164863347

2.2.2 *Feature selectors*

`\XeTeXcountselectors` ⟨*font*⟩ ⟨*feature*⟩
Expands to the number of selectors in a ⟨*feature*⟩ of a ⟨*font*⟩.

`\XeTeXselectorcode` ⟨*font*⟩ ⟨*feature code*⟩ ⟨*integer, n*⟩
Expands to the selector code for the *n*-th selector in a ⟨*feature*⟩ of a ⟨*font*⟩.

`\XeTeXselectorname` ⟨*font*⟩ ⟨*feature code*⟩ ⟨*selector code*⟩
Expands to the name corresponding to the ⟨*selector code*⟩ of a feature of a ⟨*font*⟩.

`\XeTeXisdefaultselector` ⟨*font*⟩ ⟨*feature code*⟩ ⟨*selector code*⟩
Expands to a number greater than zero if the selector of a feature of a font is on by default.

`\XeTeXfindselectorbyname` ⟨*font*⟩ ⟨*feature name*⟩ ⟨*selector name*⟩
This command provides a method to query whether a feature selector name corresponds to a selector of a specific feature contained in the font. It represents an integer corresponding to the selector number used to access the feature selector numerically. If the feature selector does not exist, the integer is `-1`.

The indices given by this command and by `\XeTeXfindfeaturebyname` can be used in Graphite fonts to select font features directly (see example below). Alternatively, they can be used as a means of checking whether a feature/selector exists before attempting to use it.

> *Example:*
>
> ```
> \font\1="Charis SIL/GR" at 10pt
> \def\featname{Uppercase Eng alternates}
> \newcount\featcount
> \featcount=\XeTeXfindfeaturebyname\1 "\featname"\relax
>
> \def\selecname{Large eng on baseline}
> \newcount\seleccount
> \seleccount=\XeTeXfindselectorbyname\1 \featcount "\selecname"\relax
> The feature selector ‘\selecname’ has index \the\seleccount
>
> \font\2="Charis SIL/GR:\featname=\selecname" at 10pt
> \font\3="Charis SIL/GR:\the\featcount=\the\seleccount" at 10pt
>
> Activating the feature: \1 ŋ \2 ŋ \3 ŋ
> ```
>
> The feature selector ‘Large eng on baseline’ has index 1
> Activating the feature: ŋ ɲ ɲ

2.2.3 *Variation axes*

`\XeTeXcountvariations` ⟨*font*⟩
Expands to the number of variation axes in the ⟨*font*⟩.

`\XeTeXvariation` ⟨*font*⟩ ⟨*integer, n*⟩
Expands to the variation code for the *n*-th feature in the ⟨*font*⟩.

`\XeTeXvariationname` ⟨*font*⟩ ⟨*variation code*⟩
Expands to the name corresponding to the ⟨*feature code*⟩ in the ⟨*font*⟩.

`\XeTeXvariationmin` ⟨*font*⟩ ⟨*variation code*⟩
Expands to the minimum value of the variation corresponding to the ⟨*variation code*⟩ in the ⟨*font*⟩.

`\XeTeXvariationmax` ⟨*font*⟩ ⟨*variation code*⟩
Expands to the maximum value of the variation corresponding to the ⟨*variation code*⟩ in the ⟨*font*⟩.

`\XeTeXvariationdefault` ⟨*font*⟩ ⟨*variation code*⟩
Expands to the default value of the variation corresponding to the ⟨*variation code*⟩ in the ⟨*font*⟩.

`\XeTeXfindvariationbyname` ⟨*font*⟩ ⟨*variation name*⟩
An integer corresponding to the internal index corresponding to the ⟨*variation name*⟩. This index cannot be used directly but may be used to error-check that a specified variation name exists before attempting to use it.

2.3 *Maths fonts*

The primitives described following are extensions of TEX 's 8-bit primitives.

In the following commands, ⟨*fam.*⟩ is a number (0–255) representing font to use in maths. ⟨*math type*⟩ is the 0–7 number corresponding to the type of math symbol; see a TEX reference for details.

Before version 0.9999.0 the following primitives had `\XeTeX` prefix instead of `\U`, the old names are deprecated and will be removed in the future.

`\Umathcode` ⟨*char slot*⟩ [=] ⟨*math type*⟩ ⟨*fam.*⟩ ⟨*glyph slot*⟩
Defines a maths glyph accessible via an input character. Note that the input takes *three* arguments unlike TEX 's `\mathcode`.

`\Umathcodenum` ⟨*char slot*⟩ [=] ⟨*math type/fam./glyph slot*⟩
Pure extension of `\mathcode` that uses a 'bit-packed' single number argument. Can also be used to extract the bit-packed mathcode number of the ⟨*char slot*⟩ if no assignment is given.

`\Umathchar` ⟨*math type*⟩ ⟨*fam.*⟩ ⟨*glyph slot*⟩
Typesets the math character in the ⟨*glyph slot*⟩ in the family specified.

`\Umathcharnum` ⟨*type/fam./glyph slot*⟩
Pure extension of `\mathchar` that uses a 'bit-packed' single number argument. Can also be used to extract the bit-packed mathcode number of the ⟨*char slot*⟩ if no assignment is given.

`\Umathchardef` ⟨*control sequence*⟩ [=] ⟨*math type*⟩ ⟨*fam.*⟩ ⟨*glyph slot*⟩
Defines a maths glyph accessible via a control sequence.

`\Umathcharnumdef` ⟨*control sequence*⟩ [=] ⟨*type/fam./glyph slot*⟩
Defines a control sequence for accessing a maths glyph using the 'bit-packed' number output by, e.g., `\Umathcodenum`. This would be used to replace legacy code such as `\mathchardef\foo=\mathcode`\`.

`\Udelcode` ⟨*char slot*⟩ [=] ⟨*fam.*⟩ ⟨*glyph slot*⟩
Defines a delimiter glyph accessible via an input character.

`\Udelcodenum` ⟨*char slot*⟩ [=] ⟨*fam./glyph slot*⟩
Pure extension of `\delcode` that uses a 'bit-packed' single number argument. Can also be used to extract the bit-packed delcode number of the ⟨*char slot*⟩ if no assignment is given.

`\Udelimiter` ⟨*math type*⟩ ⟨*fam.*⟩ ⟨*glyph slot*⟩
Typesets the delimiter in the ⟨*glyph slot*⟩ in the family specified of either ⟨*math type*⟩ 4 (opening) or 5 (closing).

`\Umathaccent` `[keyword]` ⟨*math type*⟩ ⟨*fam.*⟩ ⟨*glyph slot*⟩
Typesets the math accent character in the ⟨*glyph slot*⟩ in the family specified. Starting from version 0.9998, `\Umathaccent` accepts optional keyword:

`fixed` Don't stretch the accent, the default is to stretch it: $\widehat{M}$ vs $\hat{M}$.
`bottom` Place the accent below its base. Can be followed by the `fixed` keyword.

`\Uradical` ⟨*fam.*⟩ ⟨*glyph slot*⟩
Typesets the radical in the ⟨*glyph slot*⟩ in the family specified.

3 Character classes

The idea behind character classes is to define a boundary where tokens can be added to the input stream without explicit markup. It was originally intended to add glue around punctuation to effect correct Japanese typesetting. This feature can also be used to adjust space around punctuation for European traditions. The general nature of this feature, however, lends it to several other useful applications including automatic font switching when small amounts of another language (in another script) is present in the text.

`\XeTeXinterchartokenstate`
Counter. If positive, enables the character classes functionality.

`\newXeTeXintercharclass` ⟨*control sequence*⟩
Allocates a new interchar class and assigns it to the ⟨*control sequence*⟩ argument.

`\XeTeXcharclass` ⟨*char slot*⟩ [=] ⟨*interchar class*⟩
Assigns a class corresponding to ⟨*interchar class*⟩ (range 0–255) to a ⟨*char slot*⟩. Most characters are class 0 by default. Class 1 is for CJK ideographs, classes 2 and 3 are CJK punctuation. The boundary of a text string is considered class 255, wherever there is a boundary between a 'run' of characters and something else — glue, kern, math, box, etc. Special case class 256 is ignored; useful for diacritics so I'm told.

`\XeTeXinterchartoks` ⟨*interchar class 1*⟩ ⟨*interchar class 2*⟩ [=] {⟨*token list*⟩}
Defines tokens to be inserted between ⟨*interchar class 1*⟩ and ⟨*interchar class 2*⟩ (in that order).

Example:

```
\XeTeXinterchartokenstate = 1
\newXeTeXintercharclass \mycharclassa
\newXeTeXintercharclass \mycharclassA
\newXeTeXintercharclass \mycharclassB
\XeTeXcharclass `\a \mycharclassa
\XeTeXcharclass `\A \mycharclassA
\XeTeXcharclass `\B \mycharclassB

% between "a" and "A":
\XeTeXinterchartoks \mycharclassa \mycharclassA = {[\itshape}
\XeTeXinterchartoks \mycharclassA \mycharclassa = {\upshape]}

% between " " and "B":
\XeTeXinterchartoks 255 \mycharclassB = {\bgroup\color{blue}}
\XeTeXinterchartoks \mycharclassB 255 = {\egroup}

% between "B" and "B":
\XeTeXinterchartoks \mycharclassB \mycharclassB = {.}

aAa A a B aBa BB
```

a[*A*]a A a B aBa B.B

In the above example the input text is typeset as

```
a[\itshape A\upshape]a A a \bgroup\color{blue}B\egroup aBa B.B
```

4 Encodings

`\XeTeXinputnormalization` ⟨*Integer*⟩
Specify whether XeTeX is to perform normalisation on the input text and, if so, what type of normalisation to use. See `http://unicode.org/reports/tr15/` for a description of Unicode normalisation. The ⟨*Integer*⟩ value can be:

- **0** (default) do not perform normalisation.
- **1** normalise to NFC form, using precomposed characters where possible instead base characters with combining marks.
- **2** normalise to NFD form, using base characters with combining marks instead of precomposed characters.

`\XeTeXinputencoding` ⟨*Charset name*⟩
Defines the input encoding of the following text.

`\XeTeXdefaultencoding` ⟨*Charset name*⟩
Defines the input encoding of subsequent files to be read.

5 Line breaking

`\XeTeXdashbreakstate` ⟨*Integer*⟩
Specify whether line breaks after en- and em-dashes are allowed. Off, 0, by default.

`\XeTeXlinebreaklocale` ⟨*Locale ID*⟩
Defines how to break lines for multilingual text.

`\XeTeXlinebreakskip` ⟨*Glue*⟩
Inter-character linebreak stretch

`\XeTeXlinebreakpenalty` ⟨*Integer*⟩
Inter-character linebreak penalty

`\XeTeXupwardsmode` ⟨*Integer*⟩
If greater than zero, successive lines of text (and rules, boxes, etc.) will be stacked upwards instead of downwards.

6 Graphics

Thanks to Heiko Oberdiek, Paul Isambert, and William Adams for their help with the documentation in this section.

`\XeTeXpicfile` ⟨*filename*⟩ `[ scaled` ⟨*int*⟩ `| xscaled` ⟨*int*⟩ `| yscaled` ⟨*int*⟩ `|`
`width` ⟨*dimen*⟩ `| height` ⟨*dimen*⟩ `| rotated` ⟨*decimal*⟩ `]`
Insert an image. See below for explanation of optional arguments.

`\XeTeXpdffile` ⟨*filename*⟩ `[page` ⟨*int*⟩ `] [ crop | media | bleed | trim | art ]`
`[scaled` ⟨*int*⟩ `| xscaled` ⟨*int*⟩ `| yscaled` ⟨*int*⟩ `| width` ⟨*dimen*⟩ `|`
`height` ⟨*dimen*⟩ `| rotated` ⟨*decimal*⟩ `]`
Insert (pages of) a PDF. See below for explanation of optional arguments.

In the graphic/PDF commands above, ⟨*filename*⟩ is the usual file name argument of `\input`, `\openin`, *etc*. It must not terminated by `\relax` if options are given. ⟨*int*⟩ and ⟨*dimen*⟩ are the usual integer or dimen specifications of regular TEX.

The rotation is specified in degrees (*i.e.*, an input of '`360`' is full circle) and the rotation is counterclockwise. The syntax of ⟨*decimal*⟩ require some explanation:

⟨*decimal*⟩ → ⟨*optional signs*⟩⟨*unsigned decimal*⟩
⟨*unsigned decimal*⟩ → ⟨*normal decimal*⟩ | ⟨*coerced dimen*⟩ | ⟨*internal dimen*⟩
⟨*normal decimal*⟩ → ⟨*normal integer*⟩ | ⟨*decimal constant*⟩

A ⟨*coerced dimen*⟩ or ⟨*internal dimen*⟩ is interpreted as number with unit '`pt`'. For example, for a rotation specified with a dimension `\testdim`,

- `\testdim=45pt` results in a rotation of 45°,
- `\testdim=1in` is 72.27°, and
- `\testdim=100sp` is (100/65536)°.

In all cases the resulting decimal number for rotation x must be within the limits $-16384 < x < 16384$.

The `\XeTeXpdffile` command takes one more optional argument for specifying to which 'box' the PDF should be cropped before inserting it (the second optional argument listed in thes syntax of `\XeTeXpdffile` above). The PDF standard defines a number of (rectangular) bounding boxes that may be specified for various purposes. These are described in the PDF Standard[5] and summarised below.

media the box defining the physical page size.

[5] Adobe Systems Incorporated, 2008:
`http://www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf`

crop	the box of the page contents for display/printing purposes.
bleed	the box containing the page contents plus whatever extra space required for printing purposes.
trim	the box of the finished page after trimming the printed 'bleed box'.
art	the box containing the 'meaningful content' of the page. This could be the crop box with boilerplate text/logos trimmed off.

When not specified in the PDF to be inserted, the crop box defaults to the media box, and the bleed, trim, and art boxes default to the crop box.

`\XeTeXpdfpagecount` ⟨*filename*⟩
Expands to the number of pages in a PDF file.

7 *Cross-compatibility with pdfTEX and/or LuaTEX*

`\pdfpageheight` ⟨*dimension*⟩
The height of the PDF page.

`\pdfpagewidth` ⟨*dimension*⟩
The width of the PDF page.

`\pdfsavepos`
Saves the current location of the page in the typesetting stream.

`\pdflastxpos`
Retrieves the horizontal position saved by `\pdfsavepos`.

`\pdflastypos`
Retrieves the vertical position saved by `\pdfsavepos`.

`\ifincsname...(\else...)\fi`
TEX conditional to branch true if the expansion occurs within `\csname ... \endcsname`.

> *Example:*
>
> ```
> \def\x{\ifincsname y\else hello\fi}
> \def\y{goodbye}
> \x/\csname\x\endcsname
> ```
>
> hello/goodbye

`\ifprimitive` ⟨*control sequence*⟩ `...(\else...)\fi`
TEX conditional to test if a control sequence is a primitive and that it has not been redefined.

`\primitive` ⟨*control sequence*⟩
If the control sequence is a primitive that's been redefined, this command causes it to expand with its original (i.e., primitive) definition.

`\shellescape`
Read-only status indicating the level of 'shell escape' allowed. That is, whether commands are allowed to be executed through `\write18{...}`. Expands to zero for off; one for on (allowed); two is 'restricted' (default in TeX Live 2009 and greater) in which a subset of commands only are allowed.

> *Example:*
>
> ```
> Shell escape \ifnum\shellescape>0 is \else is not \fi enabled.
> ```
>
> Shell escape is enabled.

`\strcmp` ⟨*arg one*⟩ ⟨*arg two*⟩
Compares the full expansion of the two token list arguments. Expands to zero if they are the same, less than one if the first argument sorts lower (lexicographically) than the second argument, and greater than one if vice versa.

> *Example:*
>
> ```
> 'a' is less than 'z': \strcmp{a}{z}
>
> \def\z{a}
> The tokens expand before being compared: \strcmp{a}{\z}
>
> \def\a{z}
> Therefore, |\a| is greater than |\z|: \strcmp{\a}{\z}
>
> \edef\b{\string b}
> Also note that catcodes are ignored: \strcmp{b}{\b}
> ```
>
> 'a' is less than 'z': -1
> The tokens expand before being compared: 0
> Therefore, \a is greater than \z: 1
> Also note that catcodes are ignored: 0

`\suppressfontnotfounderror` ⟨*integer*⟩
When set to zero (default) if a font is loaded that cannot be located by XeTeX, an error message results and typesetting is halted. When set to one, this error message is suppressed and the font control sequence being defined is set to `\nullfont`.

> *Example:*
>
> ```
> \suppressfontnotfounderror=1
> \font\x="ImpossibleFont" at 10pt
> \ifx\x\nullfont
>   \font\x="Georgia" at 10pt
> \fi
> \x This would be ‘ImpossibleFont’, if it existed.
> ```

7.1 Character protrusion

`\XeTeXprotrudechars` ⟨*integer*⟩
Equivalent to `\pdfprotrudechars` in pdfTeX for controlling character protrusion or `margin kerning'. When set to zero (default), character protrusion is turned off. When set to one, it is activated but will not affect line-breaking. When set to two, line-breaking decisions will change as a result of the character protrusion.

> *Example:*
>
> ```
> \XeTeXprotrudechars=2
> \font\rm="[texgyrepagella-regular.otf]"\relax
> \rm
> \rpcode\font\XeTeXcharglyph\hyphenchar\font=250
> \hsize=20mm
> a a a a a a a a abbabbabb aabbabbabb abbabb
> ```
>
> a a a a a a
> a a a abbab-
> babb aabbab-
> babb abbabb

See the pdfTeX documentation for further details.

`\rpcode` ⟨*font*⟩ ⟨*char slot*⟩ (integer, *n*)
Sets the right-side character protrusion value of the ⟨*char slot*⟩ in the specified ⟨*font*⟩ to $n/1000$ em. n is clipped to ± 1000.

`\lpcode` ⟨*font*⟩ ⟨*char slot*⟩ (integer, *n*)
Sets the left-side character protrusion value of the ⟨*char slot*⟩ in the specified ⟨*font*⟩ to $n/1000$ em. *n* is clipped to ± 1000.

8 Misc.

`\XeTeXversion`
Expands to a number corresponding to the XeTeX version.

`\XeTeXrevision`
Expands to a string corresponding to the XeTeX revision number.

> *Example:*
>
> ```
> The \xetex version used to typeset this document is:
> \the\XeTeXversion\XeTeXrevision
> ```
>
> The XeTeX version used to typeset this document is: 0.99991