The XeTeX reference guide
https://ctan.org/pkg/xetexref

Will Robertson Khaled Hosny Karl Berry

February 19, 2023

Introduction

This document serves as a reference for additional features of XeTeX. It is not a users' guide. Much of the functionality addressed here is provided in abstracted form in various LaTeX packages and ConTeXt modules.

The descriptions here are intended to be a reasonably complete, though terse, list of the new primitives and features of XeTeX.

For contributions (very welcome!), bug reports, etc., see links from the CTAN package page: https://ctan.org/pkg/xetexref.

License

Contents

1 The `\font` command

Traditionally, fonts were selected in TEX like this: `\font\1=`⟨*tfm name*⟩ ⟨*TEX font options*⟩, where the ⟨*TEX font options*⟩ included possibilities such as `' at 10pt'` or `' scaled 1200'`, with evident 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*⟩

Each part of this command is discussed in the following, but here is an overview, starting from the end:

- The ⟨*TEX font options*⟩ are unchanged from traditional TEX.
- The ASCII double quote character `"` (hex 0x22) surrounds any font definition using the extended syntax.
- The ⟨*font features*⟩ are an optional list of names, comma or semi-colon separated, preceded by an ASCII colon `:` (0x3a).
- The ⟨*font options*⟩ are an optional list of specifiers which can only be used with system fonts (see next).
- The ⟨*font identifier*⟩ is the only mandatory part of the above syntax. If it is given in square brackets, (e.g., `\font\1="[lmroman10-regular]"`), it is taken as a font file name. Without brackets, the name is looked up as both a file name and a system font name.

This distinction between file name lookups and system font name lookups is crucial to understanding XETEX's behavior and to writing portable documents (in short: don't use system fonts).

System font name lookups use (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 does the system font lookups 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\1="[lmroman10-regular]"      find lmroman10-regular.otf in any tree
\font\2="[/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.

1.1 Font collection files

One more special case about the ⟨*font identifier*⟩: 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 (in contrast to the font features we'll describe soon, which follow a colon outside any square brackets). E.g.,

```
\font\2="[myfont.ttc:1]"          load the second font from myfont.ttc file
```

1.2 Font options: For OS-selected fonts only

The following ⟨*font options*⟩ are only applicable when the font is selected through the operating system (i.e., without square brackets):

/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.

The following ⟨*font options*⟩ control which 'renderer' XeTeX uses to interface with the font:

/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] `https://scripts.sil.org/cms/scripts/page.php?site_id=projects&item_id=graphite_home`

1.3 *Font features: For all fonts*

The ⟨*font features*⟩ is a comma or semi-colon separated list activating or deactivating various OpenType, Graphite, or AAT font features; these 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 utility files `aat-info.tex` and `opentype-info.tex` provide lists of supported features for a given font (see comments at the top of each).

1.3.1 *Font features specific to OpenType, Graphite, or AAT*

OpenType font features are usually specified with standard tags[2] in the `\font` command. 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="[LinLibertine_RI.otf]/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.

OpenType features can be activated by default:

> *Example:*
>
> ```
> \font\antt="[AntykwaTorunska-Regular.otf]"         at 12pt \antt 0
> \font\antt="[AntykwaTorunska-Regular.otf]:+aalt=0" at 12pt \antt 0
> \font\antt="[AntykwaTorunska-Regular.otf]:+aalt=1" at 12pt \antt 0
> \font\antt="[AntykwaTorunska-Regular.otf]:+aalt=2" at 12pt \antt 0
> \font\antt="[AntykwaTorunska-Regular.otf]:+aalt=3" at 12pt \antt 0
> \font\antt="[AntykwaTorunska-Regular.otf]:+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 can and do have different names in different fonts.

> *Example:*
>
> ```
> \font\gra="[CharisSIL-Regular.ttf]/GR:Small Caps=True" at 12pt
> \gra This is the Graphite font Charis SIL with small caps.
> ```
>
> [No output for the example, since no AAT or Graphite fonts are available.

[2] `https://www.microsoft.com/typography/otspec/featuretags.htm`

1.3.2 *Features for all fonts*

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

color=*RRGGBB*[*TT*]

Triple pair of hex values to specify the colour in RGB space, with an optional value for the transparency.

> *Example:*
>
> ```
> \font\9="[lmsans10-regular.otf]:color=0000FF,mapping=tex-text"
> \9 A sans blue quoted em-dash: ``---''.
> ```
>
> A sans blue quoted em-dash: “—”.

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).

letterspace=*x*

Adds x/S space between letters in words, where S is the font size.

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 ASCII `` ``---'' `` to proper typographical glyphs “—”, and so on.

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.

1.3.3 *OpenType script and language support*

OpenType font features (and font behavior) 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.

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

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

1.3.4 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.

1.3.5 Vertical typesetting

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

2 Font primitives

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

`\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="[CharisSIL-Regular.ttf]"
\font\3="[CharisSIL-Regular.ttf]/OT"
\whattype\1 \whattype\2 \whattype\3
```

cmr10 is rendered by TEX.
"[CharisSIL-Regular.ttf]" is rendered by OpenType.
"[CharisSIL-Regular.ttf]/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="[CharisSIL-Regular.ttf]"\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: "s".

`\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="[CharisSIL-Regular.ttf]"\1
> The glyph slot in Charis SIL for the Yen symbol is:
>     \the\XeTeXglyphindex"yen" . % font-specific glyph name
> Or: \the\XeTeXcharglyph"00A5.   % 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="[CharisSIL-Italic.ttf]" 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

`\XeTeXgenerateactualtext` ⟨*integer*⟩
Controls the output of `/ActualText` entry. Default is 0. When set to 1, the `/ActualText` entry is added to the output PDF for better copy/paste and search in PDF viewers.

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

These commands apply to AAT and Graphite fonts; for other kinds of fonts, they produce an error.

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 returns 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="[CharisSIL-Regular.ttf]/GR" at 10pt
> \def\featname{Uppercase Eng alternates}
> The feature ‘\featname’ has index
> \the\XeTeXfindfeaturebyname\1 "\featname"\relax
> ```
>
> [No output for the example, since no AAT or Graphite fonts are available.

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 returns 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="[CharisSIL-Regular.ttf]/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="[CharisSIL-Regular.ttf]/GR:\featname=\selecname" at 10pt
\font\3="[CharisSIL-Regular.ttf]/GR:\the\featcount=\the\seleccount" at 10pt

Activating the feature: \1 D \2 D \3 D
```

[No output for the example, since no AAT or Graphite fonts are available.

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 *Characters*

`\char` ⟨*number*⟩
This and related TEX primitives have been extended to take any 16-bit number, that is, up to 65536. As usual, hex digits must be specific in uppercase: `\char"FF`, not `\char"ff`.

`\Uchar` ⟨*number*⟩
Expands to a character token with specified slot ⟨*number*⟩ (range 0 to 1,114,111) with category code 12. While it looks superficially like the TEX primitive `\char`, `\Uchar` is an expandable operation.

> *Example:*
> ```
> \edef\x{\Uchar`\$}
> \ttfamily\meaning\x\par
> \expandafter\ifcat\x.[`other']\fi
> ```
> ```
> macro:->$
> [`other']
> ```

`\Ucharcat` ⟨*number*⟩ ⟨*catcode*⟩
Expands to a character token with slot ⟨*number*⟩ and ⟨*catcode*⟩ specified. The values allowed for ⟨*catcode*⟩ are: 1–4, 6–8 and 10–13.

> *Example:*
>
> ```
> \edef\x{\Ucharcat`\# 3}
> \ttfamily\meaning\x\par
> \expandafter\ifcat\x$[`mathshift']\fi
> ```
>
> ```
> macro:->#
> [`mathshift']
> ```

3.1 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 0–4095) 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 4095, wherever there is a boundary between a 'run' of characters and something else — glue, kern, math, box, etc. Special case class 4096 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 `https://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`, ifnextchar.etcetc.. 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 a 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 final 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 the 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:
`https://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 *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 a abbabbabb aabbabbabb abbabb
> ```
>
> a 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 Cross-compatibility with other TEX engines

All of the major (non-frozen) TEX engines support most of the functionality described in this section, in one way or another, so that macro formats (e.g., LATEX) can make use of it. Please see the pdfTEX manual for details (where all these primitive names begin with `\pdf...`).

8.1 Geometry

`\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`.

8.2 Programming

`\expanded` ⟨*general text*⟩
Carries out full expansion of a token list like `\message`, but it is still expandable.

`\ifincsname...[\else...]\fi`
TEX conditional, 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 which still has its primitive meaning, i.e., has not been redefined or undefined.

`\partokencontext` ⟨*integer*⟩
Specifies the cases in which `\par`, or the given `\partokenname` (see next) is internally emitted. The default is zero, yielding the standard behavior. See the pdfTEX manual for more.

`\partokenname` ⟨*control sequence*⟩
By default, TEX emits a control sequence `\par` at blank lines and other situations. This command changes the name of what is emitted to the given control sequence. See the pdfTEX manual for more.

`\primitive` ⟨*control sequence*⟩
This command executes the primitive meaning of the following control sequence, regardless of whether the control sequence has been redefined or made undefined. If the primitive was expandable, `\primitive` expands also. On the other hand, if the control sequence never was a primitive, nothing happens and no error is raised.

`\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.

`\showstream` ⟨*integer*⟩
If this primitive parameter has a value corresponding to an open output stream (which has been opened with `\openout`), then any `\show`, `\showthe`, `\showbox` or `\showlists` commands do not write output to the terminal, but instead write to only the referenced output stream, as if they were written with `\immediate\write`.

`\special shipout` ⟨*token list*⟩
With the additional "keyword" `shipout`, expansion of the ⟨*token listr*⟩ is delayed until `\shipout`, like non-`\immediate\write`.

`\strcmp` ⟨*arg one*⟩ ⟨*arg two*⟩
Compares the full expansion of the two token list arguments. Expands to zero if they are the same, -1 if the first argument sorts lower (lexicographically) than the second argument, and 1 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

`\tracingstacklevels` ⟨*integer*⟩
If this integer parameter is positive, and `\tracingmacros` is also positive, a prefix indicating the macro expansion depth is output on each relevant log line (e.g., `~...` at depth 2). Also, macro logging is truncated at a depth $\geq$ the parameter value.

8.3 Randomness

`\normaldeviate` ⟨*number*⟩
Generate a normally-distributed random integer with a mean of 0 and standard deviation 65 536. That is, about 68% of the time, the result will be between −65 536 and 65 536 (one standard deviation away from the mean). About 95% of results will be within two standard deviations, and 99.7% within three.

`\randomseed`
A read-only integer which allows querying of the random seed.

`\setrandomseed` ⟨*number*⟩
Set the random seed to a specific value, allowing you to replay sequences of semi-randoms at a later moment.

`\uniformdeviate` ⟨*number*⟩
Generate a uniformly-distributed random integer value between 0 (inclusive) and ⟨number⟩ (exclusive).

`SOURCE_DATE_EPOCH` (environment variable)
If this environment variable is set, its value is used as the current time in seconds since the usual Unix "epoch": the beginning of 1970-01-01, UTC. Negative or non-integer values cause a fatal error. Thus, a value of 32 would result in a `/CreationDate` and `/ModDate` values of `19700101000032Z`. This is useful for reproducible builds of documents.

`FORCE_SOURCE_DATE` (environment variable)
If this is set to `1`, XeTeX's time-related primitives are also initialized from the value of `SOURCE_DATE_EPOCH`. These primitives are `\year`, `\month`, `\day`, and `\time`. If `SOURCE_DATE_EPOCH` is not set, setting `FORCE_SOURCE_DATE` has no effect. (See the pdfTeX manual for pathological cases.)

8.4 Timing

`\elapsedtime`
Return a number that represents the time elapsed from the moment of the start of the run or the last point at which the timer was reset. The elapsed time is returned in 'scaled seconds', meaning seconds divided by 65 536. If the time exceeds 32 767 seconds, the constant value $2^{31} - 1$ will be returned.

`\resettimer`
Reset the internal timer to 0.

8.5 File handling

`\filedump [offset]` ⟨*number*⟩ `[length]` ⟨*number*⟩ `{` ⟨*filename*⟩ `}`
Expands to a hexadecimal dump of the contents of the file, starting from position zero and extending to the end of the file unless either an offset or length are given.

`\filemoddate {` ⟨*filename*⟩ `}`
Expands to the date the file was last modified in the format `D:YYYYMMDDHHMMSS+ZZZZ` as a string.

`\filesize {` ⟨*filename*⟩ `}`
Expands to the size of the file in bytes.

`\mdfivesum [file] {` ⟨*text*⟩ `}`
Expands to the MD5 sum of the ⟨*text*⟩, or if the `file` keyword is given, the MD5 sum of the contents of the file named ⟨*text*⟩.

`\input {` ⟨*filename*⟩ `}`
As of TeX Live 2020, the `\input` primitive in all TeX engines, including XeTeX, now also accepts a group-delimited filename argument, as a system-dependent extension, as in `\input{foo.tex}`. The usage of `\input` with a standard space/token-delimited filename is completely unchanged.

This group-delimited argument was previously implemented in LuaTeX; now it is available in all engines. ASCII double quote characters (`"`) are removed from the filename, but it is otherwise left unchanged after tokenization.

This extension is unlike most others in that it affects a primitive in standard TeX (`\input`), rather than being related to a new primitive, command line option, etc. This is allowed because additional methods of recognizing filenames are explicitly mentioned in `tex.web` as acceptable system-dependent extensions.

Incidentally, this does not directly affect LaTeX's `\input` command (which takes a group-delimited argument), as that is a macro redefinition of the `\input` primitive.

8.6 Fonts

`\pdfmapfile {` ⟨*map file*⟩ `}`
Defined in `xe(la)tex.ini` to place a whatsit special for `xdvipdfmx` to set a font map file. See the pdfTeX user manual for full details.

`\pdfmapline` { ⟨*line from map file*⟩ }
Defined in `xe(la)tex.ini` to place a whatsit special for `xdvipdfmx` to set up a font map. See the pdfTEX user manual for full details.

`\tracinglostchars` ⟨*integer*⟩
When set to 3 or larger, a character missing from a font provokes a full error, not just a diagnostic message (perhaps only in the log file, depending on `\tracingonline`. Also, the missing character code is always reported in hex, two digits for normal TEX fonts or `U+....` format for system fonts, in addition to its ASCII character, if printable.

`\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="[texgyrepagella-regular.otf]" at 10pt
> \fi
> \x This would be ‘ImpossibleFont’, if it existed.
> ```
>
> This would be ‘ImpossibleFont’, if it existed.

8.7 Invoking XETEX

XETEX has many command line options. They are almost all inherited from the common framework for TEX engines as implemented in Web2C (its manual is available at `https://tug.org/web2c`).

The main exceptions are `-no-pdf`, which tells XETEX to output its XDV (extended DVI) without further processing, and `-output-driver=`⟨*cmd*⟩, which processes the XDV output with ⟨*cmd*⟩; the default is `xdvipdfmx`.

9 Engine version

`\XeTeXversion`
Expands to a number corresponding to the XETEX version: 0.

`\XeTeXrevision`
Expands to a string corresponding to the XeTeX revision number: .999995.

> *Example:*
>
> ```
> The \xetex version used to typeset this document is:
> \the\XeTeXversion\XeTeXrevision
> ```
>
> The XeTeX version used to typeset this document is: 0.999995