OpenType layout features (OTL)

> OpenType layout features were introduced in mPDF >= 6.0

Advanced Typography

Many TrueType fonts contain OpenType Layout (OTL) tables. These Advanced Typographic tables contain additional information that extend the capabilities of the fonts to support high-quality international typography:

- OTL fonts support ligatures, positional forms, alternates, and other substitutions.
- OTL fonts include information to support features for two-dimensional positioning and glyph attachment.
- OTL fonts contain explicit script and language information, so a text-processing application can adjust its behavior accordingly.

mPDF 6 introduces the power and flexibility of the OpenType Layout font model into PDF. mPDF supports GSUB, GPOS and GDEF tables for now. mPDF does *not* support BASE and JSTF at present.

Other mPDF features to enhance complex scripts:

- Bidirectional (Bidi) algorithm for right-to-left (RTL) text
- support for Kashida for justification of arabic scripts
- partial support for CSS3 optional font features e.g. font-feature-settings, font-variant
- improved "autofont" capability to select fonts automatically for any script
- support for CSS :lang selector
- dictionary-based line-breaking for Lao, Thai and Khmer (U+200B is also supported)
- separate algorithm for Tibetan line-breaking

> **Note:** There are other smart-font technologies around to deal with complex scripts, namely Graphite fonts (SIL International) and Apple Advanced Typography (AAT by Apple/Mac). mPDF 6 does not support these.

What can OTL Fonts do?

Support for OTL fonts allows the faithful display of almost all complex scripts:

- Arabic (?????? ?????), Hebrew (????), Syriac (????????)
- Indic - Bengali (???????????), Devanagari (??????), Gujarati (??????), Punjabi (??? ???? ????), Kannada (??????), Malayalam (??????), Oriya (??????), Tamil (???????), Telugu (????????)
- Sinhala (????????), Thai (??????), Lao (???????), Khmer (????????), Myanmar (???????), Tibetan (??????????????????)

OTL features allow:

- Joining and Reordering
- Complex syllables
- Ligatures
- Language-dependent substitutions
- Font features - Optional substitutions Stylistic Alternatives (salt)
- CSS control of discretionary OTL features
- Mark repositioning (and diacritics)
- Mark repositioning (and Contextual substitution)
- Complex Typography An example which utilises many different GSUB and GPOS features together - first without GSUB and GPOS:
- Text Justification using Kashida

A full list of feature tags is at http://www.microsoft.com/typography/otspec/featurelist.htm

In mPDF, the following features are on by default:

- GSUB features: `locl ccmp pref blwf abvf pstf pres abvs blws psts haln rlig calt liga clig mset` (all scripts)

- GSUB features: `isol fina fin2 fin3 medi med2 init nukt akhn rphf rkrf half vatu cjct cfar` (for appropriate scripts e.g. Indic, Arabic)
- GPOS features: `abvm blwm mark mkmk curs cpsp dist requ` [`kern`]

NB '`requ`' is not listed in the Microsoft registry of Feature tags; however it is found in the Arial Unicode MS font (it repositions the baseline for punctuation in Kannada script).

`kern` is used in some fonts to reposition marks etc. and is essential for correct display, so in mPDF `kern` is on by default when any non-Latin script is used.

Complex scripts require a "shaping engine" to re-order glyphs and apply the OTL features by syllable. MS Word and Wordpad run on the Windows platform use "Uniscribe", whereas some browsers such as FireFox and OpenOffice use Pango/HarfBuzz. The different shaping engines (and indeed different versions of them) can produce different results.

Different applications have different defaults (on/off) for some of the features e.g. kerning.

When testing mPDF, if text does not appear as you expect, ensure that the font is installed on your computer, and view the HTML in a browser. Also try copying/pasting the text into Wordpad/Word/OpenOffice and ensure that the correct font has been applied.

Note that Wordpad sometimes substitutes a different font if it does not like the one you have chosen, and does not even indicate that the substitution has occurred.

CSS control of font features

See http://www.w3.org/TR/css3-fonts/#font-rend-props for information about CSS3 and font-features.

The following are supported in mPDF:

- `font-variant-position`
- `font-variant-caps`
- `font-variant-ligatures`
- `font-variant-numeric`
- `font-variant-alternates` - Only [`normal | historical-forms`] supported (i.e. most are NOT supported)
  e.g. `stylistic, styleset, character-variant, swash, ornaments, annotation` (use `font-feature-settings` for these)
- `font-variant` - as above, and except for: `east-asian-variant-values, east-asian-width-values, ruby`
- `font-language-override`
- `font-feature-settings`

`font-variant-east-asian` is NOT supported

NB `@font-face` is NOT supported

NB `@font-feature-values` is NOT supported

Note: `font-variant` specifies a single property in CSS2, whereas in CSS3 it has become a shorthand for all the other `font-variant-*` properties. `font-variant: small-caps` was the form supported in CSS2, and will work in mPDF.

See notes later about font kerning.

Examples:

```
/* use small-cap alternate glyphs */
.smallcaps { font-feature-settings: "smcp" on; }

/* convert both upper and lowercase to small caps (affects punctuation also) */
.allsmallcaps { font-feature-settings: "c2sc", "smcp"; }

/* enable historical forms */
.hist { font-feature-settings: "hist"; }

/* disable common ligatures, usually on by default */
.noligs { font-feature-settings: "liga" 0; }

/* enable tabular (monospaced) figures */
td.tabular { font-feature-settings: "tnum"; }

/* enable automatic fractions */
.fractions { font-feature-settings: "frac"; }

/* use the second available swash character */
.swash { font-feature-settings: "swsh" 2; }
```

```
/* enable stylistic set 7 */
.fancystyle {
font-family: Gabriola; /* available on Windows 7, and on Mac OS */
font-feature-settings: "ss07";
}
```

How to use OTL in mPDF

In `config_fonts.php` there are 2 variables which affect OTL features for each font family e.g.:

```
"dejavusanscondensed" => array(
    'R' => "DejaVuSansCondensed.ttf",
    'B' => "DejaVuSansCondensed-Bold.ttf",
    'I' => "DejaVuSansCondensed-Oblique.ttf",
    'BI' => "DejaVuSansCondensed-BoldOblique.ttf",
    'useOTL' => 0xFF,
    'useKashida' => 75,
),
```

useOTL
useOTL should be set to an integer between 0 and 255. Each bit will enable OTL features for a different group of scripts:

Bit	dec	hex	Enabled
1	1	0x01	GSUB/GPOS - Latin script
2	2	0x02	GSUB/GPOS - Cyrillic script
3	4	0x04	GSUB/GPOS - Greek script
4	8	0x08	GSUB/GPOS - CJK scripts (excluding Hangul-Jamo)
5	16	0x10	(Reserved)
6	32	0x20	(Reserved)
7	64	0x40	(Reserved)
8	128	0x80	GSUB/GPOS - All other scripts (including all RTL scripts, complex scripts etc)

Setting `useOTL` to 0 (or omitting it) will disable all OTL features. Setting `useOTL` to 255 or 0xFF will enable OTL for all scripts. Setting `useOTL` to 0x82 will enable OTL features for Cyrillic and complex scripts.

In a font like Free Serif, it may be useful to enable OTL features for complex scripts, but disable OTL for Latin scripts (to save processing time). However, see above - this may disable kerning in Latin scripts in certain circumstances.

useKashida
useKashida should be set for arabic fonts if you wish to enable text justification using kashida. The value should be an integer between 0 and 100 and represents the percentage of additional space required to justify the text on a line as a ratio of kashida/inter-word spacing.

Choosing fonts to add to mPDF 6

Fonts with OTL need to have GDEF, GSUB and GPOS tables in the font file. Although TrueType font files are binary files, the table names and script/feature tags are written as ASCII characters; open the .ttf or .otf file in a text editor such as Windows Notepad, and you will see GDEF, GSUB and GPOS in the first few lines if they are present. You can also search the file to see if the script tags are present for your desired scripts cf. http://www.microsoft.com/typography/otspec/scripttags.htm.

Note: The OTL specification for Indic fonts was updated in 2005 to version 2. The v2 script tag for Bengali is "bng2" whereas prior to this it was "beng". Many open-source font files are still written for the old specification. This is supported by mPDF, although v2 fonts give better results.

Note: mPDF does not support Graphite or AAT font features.

Configuring new fonts for mPDF 6
To add a font, first copy the font file to the `/ttfonts/` folder.

Then edit `config_fonts.php` to add. See the manual for details if you are not already familiar with this.

If you wish to use this font with `autoLangToFont`, you also need to edit `config_lang2fonts.php`

Setting OTL use at runtime
mPDF caches some font information in the `/ttfontdata/` folder to improve performance. This is regenerated if you change the value of `useOTL` for a font.

There may be circumstances when you wish to use OTL features with different scripts depending on the document e.g. for everyday use you may want to disable OTL for FreeSerif to save processing time, but on occasions use OTL for Indic and/or Arabic scripts. The recommended way to do this is to create 2 instances of the font e.g. in `config_fonts.php`:

```
"freeserif" => array(
'R' => "FreeSerif.ttf",
'B' => "FreeSerifBold.ttf",
'I' => "FreeSerifItalic.ttf",
'BI' => "FreeSerifBoldItalic.ttf",
'useOTL' => 0x00,
),
"freeserif2" => array(
'R' => "FreeSerif.ttf",
'B' => "FreeSerifBold.ttf",
'I' => "FreeSerifItalic.ttf",
'BI' => "FreeSerifBoldItalic.ttf",
'useOTL' => 0xFF, /* Uses OTL for all scripts */
'useKashida' => 75,
),
```

You could then either use this second font name in your stylesheets e.g.

```
<p style="font-family:freeserif2;">Hallo World (in Arabic)</p>
```

or, you could use font translation e.g.

```
$mpdf->fonttrans['freeserif'] = 'freeserif2';
```

Printed on Wed 05 Aug 2015 12:04:15 GMT +0100 (DST)