

**NAME**

Font::TTF::OS\_2 – the OS/2 table in a TTF font

**DESCRIPTION**

The OS/2 table has two versions and forms, one an extension of the other. This module supports both forms and the switching between them.

**INSTANCE VARIABLES**

No other variables than those in table and those in the standard:

```
Version
xAvgCharWidth
usWeightClass
usWidthClass
fsType
ySubscriptXSize
ySubscriptYSize
ySubscriptXOffset
ySubscriptYOffset
ySuperscriptXSize
ySuperscriptYSize
ySuperscriptXOffset
ySuperscriptYOffset
yStrikeoutSize
yStrikeoutPosition
sFamilyClass
bFamilyType
bSerifStyle
bWeight
bProportion
bContrast
bStrokeVariation
bArmStyle
bLetterform
bMidline
bXheight
ulUnicodeRange1
ulUnicodeRange2
ulUnicodeRange3
ulUnicodeRange4
achVendID
fsSelection
usFirstCharIndex
usLastCharIndex
sTypoAscender
sTypoDescender
sTypoLineGap
usWinAscent
usWinDescent
ulCodePageRange1
ulCodePageRange2
xHeight
CapHeight
defaultChar
breakChar
maxLookups
```

Notice that versions 0, 1, 2 & 3 of the table are supported. Notice also that the Panose variable has been broken down into its elements.

## METHODS

### `$t->read`

Reads in the various values from disk (see details of OS/2 table)

### `$t->out($fh)`

Writes the table to a file either from memory or by copying.

### `$t->XML_element($context, $depth, $key, $value)`

Tidies up the hex values to output them in hex

### `$t->XML_end($context, $tag, %attrs)`

Now handle them on the way back in

### `$t->minsize()`

Returns the minimum size this table can be. If it is smaller than this, then the table must be bad and should be deleted or whatever.

### `$t->update`

Updates the OS/2 table by getting information from other sources:

Updates the `firstChar` and `lastChar` values based on the MS table in the cmap.

Updates the `sTypoAscender`, `sTypoDescender` & `sTypoLineGap` to be the same values as `Ascender`, `Descender` and `Linegap` from the `hhea` table (assuming it is dirty) and also sets `usWinAscent` to be the sum of `Ascender+Linegap` and `usWinDescent` to be the negative of `Descender`.

### `$t->guessRangeBits (\%map, [%cp_threshold, [%u_threshold]])`

Set the `ulCodePageRange` and `ulUnicodeRange` fields based on characters actually present in the font.

`%map` is a hash keyed by USV returning non-zero for characters present (e.g. use `{'val'}` a from Unicode cmap).

The two optional parameters are percentage of characters within the codepage or unicode range that need to be present to constitute coverage. A threshold of 0 causes corresponding range bits to be set if any characters are present at all, while a negative value causes the corresponding range bits to be unchanged. Defaults are 50 and 0, respectively.

For codepage bits, the threshold is percentage of characters between 0xC0 and 0xFF that need to be present to constitute coverage). For codepages other than 1252, characters (e.g., punctuation) that are defined identically to cp1252 are ignored for the purposes of this percentage calculation. Looks only for SBCS codepages, not DBCS.

For Unicode range bits that represent multiple ranges, e.g., bit 29 represents:

Latin Extended Additional	1E00-1EFF
Latin Extended-C	2C60-2C7F
Latin Extended-D	A720-A7FF

the bit will be set if any of these ranges meet the threshold requirement.

## BUGS

None known

## AUTHOR

Martin Hosken <<http://scripts.sil.org/FontUtils>>.

## LICENSING

Copyright (c) 1998-2016, SIL International (<http://www.sil.org>)

This module is released under the terms of the Artistic License 2.0. For details, see the full text of the license in the file LICENSE.