Determining vertical font metrics can be tricky. Setting them wrongly may result in versal accents or large ascenders being clipped or squashed in some applications. Also, applications refer to vertical font metrics for calculating default line distances, though inconsistently. But observing some rules makes it easy to define vertical font metrics.

**PREMISES**

You have designed all glyphs in your font or font family.
You have set **upm** and designed your glyphs such that ‘ideal’ ascenders and descenders, as found in b h l and g j p, fit into it and maybe leave a little space.

**SETTING VERTICAL FONT METRICS**

There are only two sets of values to be specified:

1. The ‘ideal’ ascender and descender values. These can be taken from letters like b h l and g j p, plus a little space. Indicated by the box around the green p l.

2. The ‘real’ ascender and descender values. These include the typeface’s or family’s tallest glyphs, e.g. versals with diacritic marks. Indicated by the box around the red Ä.

When determining these values, we reverse the order:

(2) In **Font Info/TrueType-specific settings**, hit ‘Recalculate’ to find a font’s maximum ascendings and descendings. (Do this for every style of a family, and proceed with largest values.) These are found in \( \text{os/2.usWinAscent} \) and \( \text{os/2.usWinDescent} \). Round them up a bit, then copy the result to \( \text{hhea.Ascender} \) and \( \text{hhea.Descender} \).

Set \( \text{hhea.LineGap} = 0 \) (zero).

---

* Read Roberts of Adobe tested some applications to see how these determine baseline-to-baseline distance and the distance from the top of the textbox to the first line. Results are quoted in John Hudson’s *Vertical Metrics How-To*: http://typophile.com/wiki/Vertical%20Metrics%20How-To

1. By setting \( \text{hhea.LineGap} = 0 \), two conditions are met at the same time, see the comment in the more detailed section (“This makes sure …”). After finding out, I got to know that Read Roberts came to the same conclusion, and that the same method was an earlier recommendation by John Hudson and is his current one again – see the link above.
In the *Font Info/Key dimensions* dialog, choose *keyDimensions.Ascender* and *keyDimensions.Descender* values so that 
*keyDimensions.Ascender + |keyDimensions.Descender| = UPM*. Copy these values to *os/2.sTypoAscender* and *os/2.sTypoDescender* in *Font Info/TrueType-specific settings*. Finally, calculate *os/2.sTypeLineGap* by 
\[
( os/2.usWinAscent + |os/2.usWinDescent| ) - ( os/2.sTypoAscender + |os/2.sTypoDescender| )
\]
(And don’t forget to copy *Font Info/Key dimensions* and *Font Info/TrueType-specific settings* to the other styles of your font family.)

Now you got three `parcels` (indicated by brackets in the screenshot) each of which should give the same sum. That’s it!

If you want a ‘default’ line distance (leading) of 120% of the type size, then make sure that *os/2.sTypoLineGap = 0.2 × upm*, and adjust *os/2.usWinAscent* and *os/2.usWinDescent* as well as *hhea.Ascender & hhea.Descender*.

**The small print**

The above method puts emphasis on avoiding squashing or clipping of glyphs. In fonts with very large ascenders or descendents, the default line distance will be accordingly large. This should be acceptable – designers will set leading manually anyway.

None of the following conditions is required by any specifications:

(a) \( os/2.sTypoAscender + |os/2.sTypoDescender| = upm \)

(b) \( os/2.usWinAscent + |os/2.usWinDescent| = upm \)

(c) \( hhea.Ascender + |hhea.Descender| = upm \)

Starting from the bottom, meeting conditions (b) and (c) may be impossible even with normally dimensioned typefaces:

As for (c), following Apple’s specifications, *hhea* values have the same purpose as *os/2.sTypo* values – defining the ‘ideal’ ascender & descender height.\(^2\) However, Microsoft’s recommendations for *hhea* values make it clear that *hhea.Ascender* and *hhea.Descender* should give values for maximum ascenders and descendents to avoid squashing of glyphs larger than these values.\(^3\)

As for (b), specifications for *os/2.usWin* values state that some applications use these values for calculating default line distance, and if these values are set too small, this may result in clipping of larger glyphs.\(^4\)

---


The Document says: ‘The values for ascent, descent and lineGap represent the design intentions of the font’s creator rather than any computed value.’ (Italics mine.)

\(^{3}\) http://www.microsoft.com/typography/otspecs/recom.htm

The chapter ‘Baseline to Baseline Distances’ (below the table in section ‘Macintosh’) says: ‘If pixels extend above the ascent or below the descent, the character will be squashed in the vertical direction so that all pixels fit within these limitations; this is true for screen display only.’ Unfortunately, the terminology is not very clear. The use of the terms ‘ascent’ and ‘descent’ would refer to *os/2.usWin* rather than to *hhea* values. But placing this sentence in the Macintosh section and speaking of squashing instead of clipping suggests that *hhea* values are meant. — Irrespective of questions of exegesis, it is a fact that some Mac applications (BBEdit, Apple Works) squash glyphs whose actual dimensions exceed *hhea.Ascender* and *hhea.Descender*. This cannot be ignored.

\(^{4}\) http://www.microsoft.com/typography/otspecs/os2.htm#wu
Condition (a) is not a must either but is recommended by Microsoft, and is met in some of their OpenType TT fonts and all Adobe OpenType PS fonts. I presuppose that a typeface’s ascenders & descenders remain within the UPM boundaries. Making the sum of os/2.sTypoAscender & os/2.sTypoDescender match the UPM, glyphs’ ascenders and descenders will not overlap in text editors which calculate baseline-to-baseline-distance from os/2.sTypoAscender & os/2.sTypoDescender alone. – Again, however, condition (a) is not required. You may easily increase these values later if you find that ascenders and descenders look better if they are longer, without worrying that both design & values do not fit into UPM any more. Even changing the UPM is possible:

It is common but not required that OpenType PS fonts have UPM = 1000, and that OpenType TT fonts have UPM = 2048. (So, if you design with PS-outlines and UPM = 1000 to generate OpenType PS fonts, you may also generate a OpenType TT version without re-scaling UPM and glyphs.) However, UPM should be a power of 2 in OpenType TT fonts.

Condition (a) is followed in this document to simplify calculations. At the same time this shall emphasize the importance of a systematic approach both to setting metrics values correctly and to designing glyphs – which both depend on each other.

**An example**

If you plan to produce OpenType PS fonts, **UPM** = 1000 suggests itself. Of course you have designed your typeface so that ascenders and descenders, with some space left, fit into the UPM.

You use ‘Recalculate’ to find that the maximum extensions are, say, 786 and -353, round them to 800 and -400, and set these for hhea.Ascender & hhea.Descender, and os/2.usWinAscent & os/2.usWinDescent respectively.

You already know the average ascender & descender heights and set keyDimensions.Ascender & keyDimensions.Descender and os/2.sTypoAscender & os/2.sTypoDescender accordingly, so these match the UPM if added: 700 + |-300|.

Then by our definition, hhea.LineGap = 0. And by calculating the difference (800 + |-400|) - (700 + |-300|) you get os/2.sTypoLineGap = 200. And almost by accident, our default line distance is 120 % of the font size.

Done!

---

5. [http://www.microsoft.com/typography/otspec/recom.htm#tad](http://www.microsoft.com/typography/otspec/recom.htm#tad)

End of the first paragraph: ‘The value of (sTypoAscender - sTypoDescender) is recommended to equal one em.’


Description for unitsPerEm: ‘This value should be a power of 2 for fonts that have TrueType outlines.’
MORE DETAILED

Key dimensions

- \( \text{keyDimensions.Ascender} = \frac{0 s}{2 s}.\text{TypoAscender} \)
- \( \text{keyDimensions.Descender} = \frac{0 s}{2 s}.\text{TypoDescender} \)
  
while (arbitrarily defined)

- \( \text{keyDimensions.Ascender} + |\text{keyDimensions.Descender}| = \frac{0 s}{2 s}.\text{TypoAscender} + |\frac{0 s}{2 s}.\text{TypoDescender}| = \text{UPM} \)

- \( \frac{0 s}{2 s}.\text{usWinAscent} = \text{hhea.Ascender} \)
- \( \frac{0 s}{2 s}.\text{usWinDescent} = |\text{hhea.Descender}| \)

These should exceed the maximum vertical glyph dimensions found in the font(s).

Furthermore, we just define: \( \text{hhea.LineGap} = \text{zero} \) by default

This makes sure that two conditions are met at once:

1. \( \frac{0 s}{2 s}.\text{usWinAscent} = \text{hhea.Ascender} \)
   and \( \frac{0 s}{2 s}.\text{usWinDescent} = |\text{hhea.Descender}| \)
2. \( \frac{0 s}{2 s}.\text{usWinAscent} + |\frac{0 s}{2 s}.\text{usWinDescent}| = \text{hhea.Ascender} + |\text{hhea.Descender}| + \text{hhea.LineGap} \)

In a last step: \( \frac{0 s}{2 s}.\text{TypoLineGap} = \text{auto, i.e.} \)

- \( \frac{0 s}{2 s}.\text{TypoLineGap} = (\text{hhea.Ascender} + |\text{hhea.Descender}|) - (\frac{0 s}{2 s}.\text{TypoAscender} + |\frac{0 s}{2 s}.\text{TypoDescender}|) \)

Thus:

- \( \frac{0 s}{2 s}.\text{TypoAscender} + |\frac{0 s}{2 s}.\text{TypoDescender}| + \frac{0 s}{2 s}.\text{TypoLineGap} = \text{hhea.Ascender} + |\text{hhea.Descender}| \)
  - \( = \frac{0 s}{2 s}.\text{usWinAscent} + |\frac{0 s}{2 s}.\text{usWinDescent}| \)

If a ‘default’ baseline-to-baseline-distance of 120% of type size (determined by \( \text{UPM} = \frac{0 s}{2 s}.\text{TypoAscender} + |\frac{0 s}{2 s}.\text{TypoDescender}| \)) is desired, these ratios are required too:

- \( \frac{0 s}{2 s}.\text{TypoLineGap} = 0.2 \times (\frac{0 s}{2 s}.\text{TypoAscender} + |\frac{0 s}{2 s}.\text{TypoDescender}|) \)
  and

- \( 1.2 \times (\frac{0 s}{2 s}.\text{TypoAscender} + |\frac{0 s}{2 s}.\text{TypoDescender}|) = \text{hhea.Ascender} + |\text{hhea.Descender}| \)
  - \( = \frac{0 s}{2 s}.\text{usWinAscent} + |\frac{0 s}{2 s}.\text{usWinDescent}| \)

This section however is less to describe how to set values by hand, but may be a step towards a simplified interface and (semi)automatic calculation of vertical font metrics under its hood ...

Values inbetween bars are to be treated as positive values for calculations. In FontLab Studio, however, all descender values should be negative and all others positive.
THE SIL APPROACH

More recently (I am writing this addendum on 17 June 2017) sil has proposed a much simpler approach.² Treat all ascender-descender(-linegap) pairs or triplets the same, fill them with what’s called ‘real’ values on page 1 of this document. As an illustration:

<table>
<thead>
<tr>
<th>TrueType-specific settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrueAscender: 1000</td>
</tr>
<tr>
<td>TrueDescender: -400</td>
</tr>
<tr>
<td>TypoLineGap: 0 (zero)</td>
</tr>
<tr>
<td>WinAscent: 0</td>
</tr>
<tr>
<td>WinDescent: +400</td>
</tr>
</tbody>
</table>

Rephrasing the description of page 1 accordingly:

Consider the Key dimensions Ascender and Descender values as independent of all those TrueType-specific settings’ ascender and descender values. In Font Info/TrueType-specific settings, hit ‘Recalculate’ to find a font’s maximum ascenders and descenders. (Do this for every style of a family, and proceed with the largest values.) The maximum ascenders and descenders are found in the os/2.usWinAscent and os/2.usWinDescent fields. Round these values up a bit so you get ‘nice’ values.

Copy these values to hhea.Ascender and hhea.Descender, and set hhea.LineGap = 0.

Copy these values to os/2.sTypoAscender and os/2.sTypoDescender, and set os/2.sTypeLineGap = 0.

This approach should be the safer bet for webfonts, too.


17 June 2017 // v 1.08

(A first version of this description was part of the Font Naming Interface document of 9. December 2005 // v 1.01.
See http://www.kltf.de/downloads/FontNamingInterface-kltf.pdf for a current version of Font Naming Interface.)

All rights reserved.

Despite of all efforts made, there is no warranty that this document is up-to-date or error-free. You use information provided in this document at your own risk. You are encouraged to double-check with official specifications and recommendations provided by Adobe & Microsoft, and test results in various applications and operating systems.

FontLab Studio’s interface as shown above is © of FontLab.
All product and company names mentioned in this document may be trademarks or registered trademarks of their respective companies.

Download URL for this document is http://www.kltf.de/downloads/FontMetrics-kltf.pdf