

Composite Data Update

FontCreator 12 brings some major changes and improvements to CompositeData.xml so this document has been rewritten instead of just appending any changes to the previous version. The data can be edited in a plain text editor (I use a WordPad clone called Jarte). Any changes to CompositeData.xml after the release of FontCreator 12 will be appended to the end of this document. Anchor-based composites usually improve the positioning of diacritics, especially for italic typefaces. Mix both methods for optimal results.

Instructions for CompositeData.xml

<InheritAdvanceWidth>48</InheritAdvanceWidth> Inherits the advance width of glyph figure zero (decimal code-point 48). Used by currency symbols and mathematical operators.

<InheritLSB>45</InheritLSB> Inherits the left side-bearing from glyph hyphen (decimal code-point 45).

<InheritRSB>116</InheritRSB> Inherits the right side-bearing from glyph t (decimal code-point 116)

<InheritSB>65</InheritSB> Inherits the side-bearings from glyph Capital A (decimal code-point 65).

<InheritFromGlyphLSB>a.pcap</InheritFromGlyphLSB>

<InheritFromGlyphRSB>b.smcp</InheritFromGlyphRSB>

<LSB>120</LSB> Sets the left side-bearing to 120 funits.

<RSB>44</RSB> Sets the right side-bearing to 44 funits.

<SB>100</SB> Sets both side-bearings to 100 funits.

<SwapSB>True</SwapSB> Swap inherited side-bearings (useful for mirrored or rotated glyphs)


<AdvanceWidth>1024</AdvanceWidth> Sets the advance width to 1024 funits. Used by diacritics. Combining diacritics, being non-spacing characters, have zero width.

The above instructions apply to the <Composite> definition, while the following apply to the composite glyph <Member> definition.

<Member id = "2"> Each glyph member can now have an **id** number for use later in that composite definition.

<Base>2</Base> By default, the first glyph member is assumed to be the base glyph on which the diacritics are positioned. This instruction tells FontCreator to use the second (or another) glyph member as the base glyph. Used for composites like Dz caron.


<Pos>Auto</Pos> For many composites, this is all that is needed. FontCreator will position most diacritics correctly over or under the base glyph, raising them for uppercase by the difference between CapHeight and x-height. If you're not satisfied with the results, you can manually change the CapHeight on Format, Settings, Metrics, and recompose the uppercase composites. Later you can recalculate the CapHeight.

 Design standard diacritics in the correct vertical position for lowercase.

<Pos>Next</Pos> For most ligatures or digraphs like IJ, Nj, and Dz, and Alphabetic Presentation Forms like fi, fl, ff, the second glyph is positioned directly after the first, in the position it would normally be without kerning if typed as a separate character. If the designer opts to kern ligatures more tightly, the advance width will need reducing to suit.

<TreatAsUppercase>TRUE</TreatAsUppercase> Tells FontCreator to regard the base glyph as a capital letter, even though it is lowercase. Use by h caron, k caron, etc.

<TreatAsLowercase>TRUE</TreatAsLowercase> Tells FontCreator to regard the base glyph as lowercase. Used by Capitals with Stacking Diacritics.

 Design stacking and uppercase diacritics in the correct vertical position for uppercase.

<InheritLSB>TRUE</InheritLSB> The composite inherits the left side bearing of the composite glyph member.

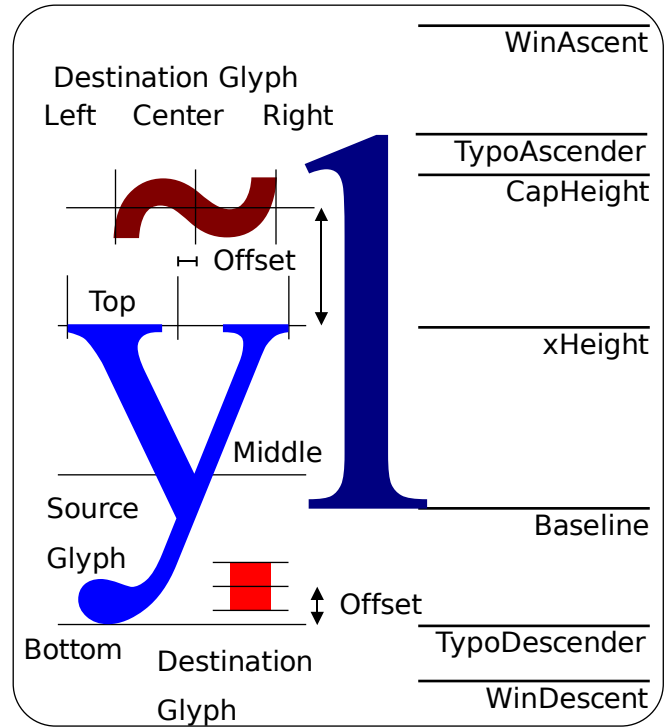
<InheritRSB>TRUE</InheritRSB> The composite inherits the right side bearing of the composite glyph member. Used together, these two ensure the correct advance width for fractions and ligatures.

<AlignHorizontal> Aligns the destination glyph horizontally with the source glyph. An offset can be used too. In the illustration, the tilde can be offset from the centre of the "y" by a percentage of the width of the "y." Positive values offset the point of alignment to the right. One could also offset the tilde by a percentage of its own width, or by a fixed value.

<Source> Defines the point on the source glyph on which the destination glyph is aligned
<Position> Left, Right, or Center (sic, not Centre) of the base glyph
</Source> Ends the definition of the source glyph
<Dest> Defines the point on the destination glyph used for alignment
<Position> Left, Right, or Center of the destination glyph that is moved
<Offset>10</Offset> Adjusts the alignment point by a fixed value
 Offsets may be a percentage (25%), or defined in units per em (0.80 em), and can be negative
</Dest> Ends the definition of the destination glyph
</AlignHorizontal> Ends the definition of one pair of glyphs to be aligned horizontally.

<AlignVertical> Aligns the destination glyph vertically with the source glyph. For vertical alignment, the source may also be any of these font metrics: WinAscent, TypoAscender, CapHeight, xHeight, Baseline, TypoDescender, or WinDescent. Vertical offsets can be a fixed value, a percentage of the glyph's height, a percentage of the font metrics, or expressed in units per em (1em = 2048 funits for many fonts). Positive values offset the point of alignment upwards.

<Source> Defines the point on the source glyph on which the destination glyph is aligned
<Position> Top, Middle, or Bottom of the source or destination glyph; or at the font's CapHeight, xHeight, Baseline, WinAscent, WinDescent, TypoAscender, or TypoDescender
</Source> Ends the definition of the source glyph
<Dest> Defines the point on the destination glyph used for alignment
<Position> Top, Middle, or Bottom of the destination glyph that is moved
<Offset>10</Offset> Adjusts the alignment point by a fixed value
 Offsets may be a percentage (e.g. 25%), or negative
</Dest> Ends the definition of the destination glyph
</AlignVertical> Ends the definition of one pair of glyphs to be aligned vertically



Other Instructions

<Composite> Start of composite definition **</Composite>** End of composite definition
<!-- Any Text Here --> Comments are used for glyph names
<GlyphMapping> Decimal code-point of composite or glyph member
<Member> Start of composite glyph member definition **</Member>** End of definition
<UseMetrics>TRUE</UseMetrics> Use this glyph's metrics. FALSE is assumed otherwise.
<XPos>100</XPos> Horizontal offset 100 funits to the right (negative values are to the left)
<YPos>500</YPos> Vertical offset 500 funits upwards (negative values are downwards)
<XScale>-1</XScale> Flips the glyph horizontally about the Y-Axis (range ± 1.9999)
<YScale>1.25</YScale> Scales the glyph by 125% vertically (range ± 1.9999)

These instructions are precise, but only for the font used to design the code. If it doesn't work well for your fonts, let us know where the problem lies, and perhaps we can improve the code. The definitions were based on the free font [Verajja Regular](#).

One can set bearings for the composite, and align glyph members horizontally or vertically with each other, or vertically with font metrics. One can use offsets from these positions — either as a percentage or as an absolute value.

Wherever CompositeData.xml uses variables that are set from font metrics, the user can make adjustments by manually changing these font metrics. After the design stage is complete, the correct metrics for the font can be recalculated.

FontCreator can use fallback glyph members if they are defined. That is, if the first glyph is unavailable, it will use another one. For example, if Uppercase diacritics are available it will use them, but if not, regular diacritics will be used instead: The end result is that after using Complete Composites, the user should have much less work to do aligning diacritics. It will still be necessary to check, but most will be right.

Processing Order for Instructions

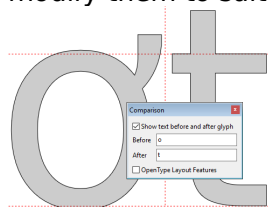
1. Add glyph members to the composite;
2. Set XPos, YPos, XScale and YScale values
3. Apply Next Pos
4. Apply Auto Pos
5. Apply horizontal and vertical alignment
6. Apply Inherit Bearings
7. Apply UseMetrics

Stacking Diacritics for Vietnamese


The limiting factor is the space available above capitals. Stacking diacritics vertically will increase the line-spacing required to maintain legibility. It is better to reduce the height of the diacritics and/or stack them side by side. Times New Roman does the former.

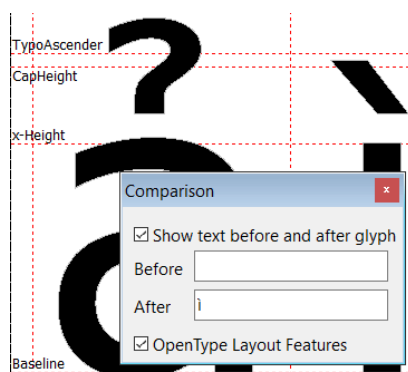
In Times New Roman, the diacritics are much smaller for Uppercase than those for lowercase. I prefer the method used by Victor Gaultney for Gentium. The diacritics for upper and lowercase are almost the same size, but at a shallower angle.

The stacking diacritics are now unmapped. Use the Glyph Transform feature to add these diacritics. For Vietnamese you will need to add the Combining Hook Above (777) and Combining Horn (795). Complete composites will approximate these diacritics with the Question Mark and Right Single Quote respectively. Convert them to simple glyphs and modify them to suit the design of the font that you are creating.



To design the combining horn, use the Comparison toolbar to display the lowercase "o" or "u," and display a letter like "t" or "v" to the right. Design the horn to avoid clashes with the letters that follow it. To design the combining hook, display an "a" before, and an diacritical vowel afterwards. This will give a guide to the size, weight, and position for the combining hook diacritic. Most combining diacritics will have zero advance width and a negative side-bearing.

 Stacking diacritics should have Contour 0 on the left, and Contour 1 or 2 on the right.



After designing the diacritics, run the Transform script to add the Vietnamese glyphs, and most of the hard work is done automatically. All you need to do is check that diacritics are in the right position. Stacking diacritics won't be moved vertically. They are designed to fit between WinAscent and the uppercase vowels.

Superscripts

Super/subscripts should be made heavier to compensate for scaling. Use the Glyph Transform Wizard to create them. Click the folder icon to open the Superscript Transform script.

This is what it does:



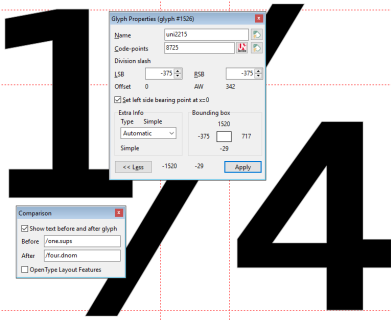
Script: Superscript.xml
Insert Characters 8304, 185, 178, 179, 8308-8316
Complete Composites
Decompose
Scale (65, 60), LT (about Top Left corner of the glyph).
Bold (26, 20)
Move (0, -20)
Width (fixed 899) Both sides
Set left side-bearing at x=0

Top alignment is designed to align with the top of the figures. Change the Move value from -20 to +80 (or more) for higher superscripts like the ² in the above illustration. The advance width (default 899) must be calculated to suit each font by scaling the figure width by 65% and adding twice the horizontal weight increase to the overall width.

Subscripts

Subscripts are composed from superscripts. This reduces the workload if you use glyph transform to increase the weight of superscripts as recommended. The subscripts are designed to bisect the baseline. Denominators for fractions can also use the superscript glyphs, but can be designed to sit on the baseline.

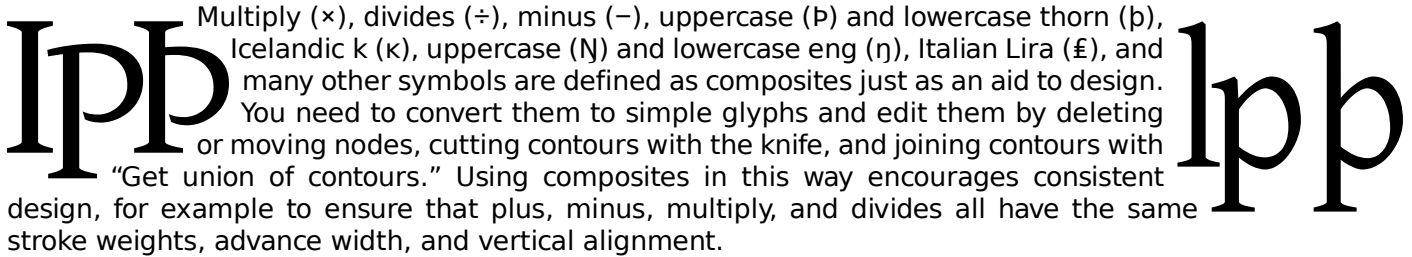
Fractions



Fractions are composed from superscripts and denominators. Both numerators and denominators use the superscript glyphs, but denominators are moved down to the baseline. If the fraction slash (8260) does not exist, compose it from forward slash, then skew it forwards. To adjust the spacing of fractions, edit the fraction slash. Type the glyph names as `/one.sups` and `/four.dnom` (or whatever you prefer) in the comparison toolbar. Move the glyph and the right side-bearing line until the spacing is about right. Then compose the fractions again. The instruction `<Pos>Next</Pos>` is used for fractions so that their advance width is simply the sum of the widths of each composite glyph member.


Composites Used to Aid Design

Multiply (\times), divides (\div), minus ($-$), uppercase (P) and lowercase thorn (þ), Icelandic k (κ), uppercase (N) and lowercase eng (η), Italian Lira (€), and many other symbols are defined as composites just as an aid to design. You need to convert them to simple glyphs and edit them by deleting or moving nodes, cutting contours with the knife, and joining contours with “Get union of contours.” Using composites in this way encourages consistent design, for example to ensure that plus, minus, multiply, and divides all have the same stroke weights, advance width, and vertical alignment.




Composites With Stroke

Composites with stroke and currency symbols use stroke overlays (821-823) to ensure a consistent stroke weight. It is often similar to other horizontal strokes in the font. These composites should be made simple before using “Get Union of Contours.” If the overlay strokes are missing, the underline glyph (95) will be used instead.




Greek and Coptic

Diacritical characters in this character set are composed from Greek alphabets. Although some fonts use the same letter forms for Greek and Latin, the design may be different. Compare Y and Upsilon, or “v” and “nu” from Gentium (illustrated on the right). Complete composites is no more than the first stage in designing the Greek alphabet. If your font contains the Basic Greek character set, you can generate the full Greek Extended character set by running the Greek Extended Transform script.



Composite data uses diacritics in the Greek Extended character set. These are composed from Latin diacritics: tilde, grave, acute, and left/right single quotes. The diacritics will need to be decomposed and adjusted to be suitable for Greek Extended. The » [Gentium](#) « font was used to design the script and composite data. If you look at that, you will see how best to design the Greek diacritics to work well with the Transform script.



Small Capitals and Petite Capitals

These are intended for use with the Unmapped Small Capitals transform script, which scales them to about 80% of Caps Height, or the Unmapped Petite Capitals transform script, which scales them to the x-height. Users should adjust the scale in the script to suit the proportions of their font. Small and Petite Capitals are no longer assigned to code-points in the Private Use Area, but use glyph names. Transform scripts from earlier versions of FontCreator can be used to add mapped glyphs for applications that do not support OpenType features.

Petite Capitals • SMALL CAPITALS

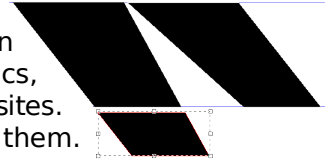
The Private Use Area (PUA) may be used as you like, e.g. for glyphs that have no assigned Unicode mapping, but FontCreator no longer needs to use it for OpenType glyphs, and it is generally recommended to avoid mapping glyphs used by OpenType features. [See this discussion on TypeDrawers.](#)


Old Style Figures

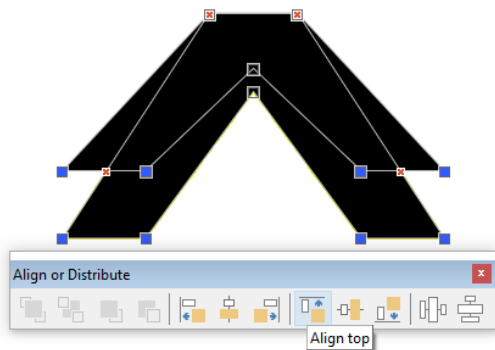
OldStyle (`onum`) or non-lining figures need to be redesigned for use with lowercase text. The tops should align with the x-height, and the descenders on figures like 3, 5, 7, and 9, should descend below the baseline. They can be added to fonts using the Unmapped OldStyle Figures Transform script. Check that they do not exceed the WinDescent or recalculate the font’s metrics if necessary. If the default figures for a font are non-lining figures one can add Lining Figures using the Unmapped Lining Figures script.

Uppercase Diacritics

If you want to use diacritics with reduced vertical height for uppercase, you can define Uppercase (.case) diacritics. Lowercase composites use the regular diacritics, but if case-sensitive diacritics are defined they will be used for uppercase composites. Use the Uppercase Glyph Transform script to insert these diacritics, then redesign them.



 Uppercase diacritics are used above Capitals, Petite Capitals, and Small Capitals.



Don't just scale the diacritics vertically. Copy them, scale the copies vertically by about 70%, then align the bottom nodes, before deleting the scaled version. Some nodes may need to be adjusted to maintain the correct weight.

Align Uppercase diacritics vertically for Capitals. Horizontal alignment is not critical as Complete Composites will align them horizontally automatically. Some manual adjustment may be needed, especially for italic type styles or for scripts.


The new anchor-based feature should give more consistent results with all kinds of different type faces.

Narrow Diacritics

If you want narrow diacritics for use with letters like i, j, or l, you can define narrow (.narrow) diacritics. Scale the regular diacritics horizontally by about 80%. Align narrow diacritics vertically for lowercase. If used in uppercase for diacritics above capital I or J, or above lowercase l, they will be moved up, whereas Uppercase (.case) diacritics will not be moved vertically, only horizontally. For smaller diacritics for use with petite capital glyphs that are not narrow, use a .cap suffix, e.g. dotbelowcomb.cap and commaaccentcomb.cap.

Modifying CompositeData.xml

This plain text file can be opened in any text editor, but if it supports line numbering, bookmarks, and regular expressions for find and replace, it will make editing the file easier.. After editing CompositeData.xml, you will need to restart Font Creator for the changes to take effect.

 Insert and design all diacritics first, then use complete composites.

Glyph Names or Glyph Mappings?


FontCreator 10 was the first version to allow the use of glyph names as well as glyph mappings. Before that, the Complete Composites feature depended on glyphs being mapped to a code-point. This is not a problem for composite glyphs that have a Unicode code-point assigned to them, but any glyphs in the Private Use Area had to occupy a particular code-point slot to allow automation of Complete Composites and Transform scripts. Small Capitals or Petite Capitals, for example shared a range from 58033-59829 (decimal). A Transform script could be used to insert nearly 200 characters mapped to these code-points, and generate glyphs from the font's Latin Basic capitals of the approximate size and weight for Small or Petite Capitals. The scripts can be edited to adjust the size and weight as needed.

This necessity of using PUA code-points caused conflicts with fonts not created in FontCreator, which have their own way of mapping extra glyphs for OpenType features. Some fonts leave them unmapped, while some use PUA code-points that might be reserved for use by FontCreator.

The Complete Composites feature in FontCreator 12 almost completely avoids mapping glyphs to the PUA. Only a few glyphs, which have no Unicode assignment, are given PUA code-points such as é (proposed by the Medieval Unicode Font Initiative).

The upside of glyphs mapped to the PUA is that they can be used in applications that do not support OpenType features. If the application lacks any method for inserting them they can be copied and pasted from Windows Character Map, or from another application that does support them.

Unmapped glyphs can only be used by Composite Glyphs or by OpenType features in applications that support them. Not mapping them protects users from inserting them — which may well cause problems on changing fonts in publications. A definition such as **<GlyphName>**gravecomb.case**<GlyphName>** can be used within CompositeData.xml to reference a combining grave diacritic for use with Uppercase composite glyphs like À grave. A code-point is no longer required. If glyphs are mapped to Uppercase diacritics they will be used in preference to regular sized diacritics over capitals. If the Uppercase diacritics are present, they will be used with petite capitals and/or small capitals.

 Unmapped definitions follow those mapped to code-points.

From the Tools menu, Glyph Names, Generate, user-friendly names should be generated. The Complete Composites feature requires them to add Unmapped Small Capitals with an OpenType tag suffix added to the base glyph's name, a.smcp ... agrave.smcp, be-cyrl.smcp, alpha.smcp, etc. Transform scripts will scale these glyphs and adjust the weight to suit the font design.

Edit the script to fine-tune the scale, weight, and position. Definitions like that below generate composites from the base glyphs and diacritics.

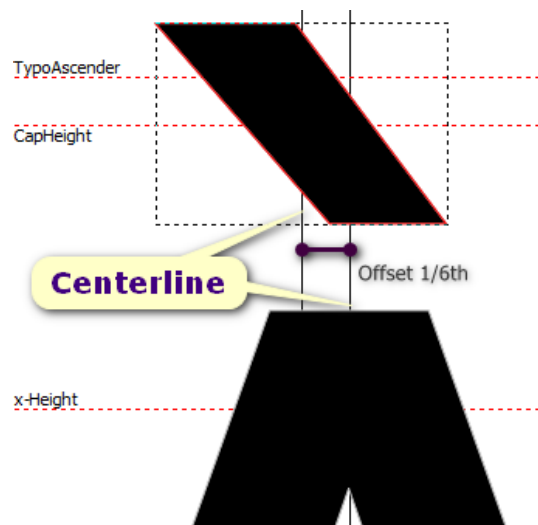
Offsets

<!-- I have added comments in angled brackets to annotate the script -->

```

<Composite><!-- Unmapped Small Capital agrave -->
<GlyphName>agrave.smcp</GlyphName> <!-- case-sensitive name of glyph in font -->
  <Member id="1"> <!-- First composite glyph member -->
    <GlyphName>a.smcp</GlyphName>
    <UseMetrics>TRUE</UseMetrics> <!-- Use the bearings and advance width of a.smcp -->
  </Member>
  <Member id="2"> <!-- Second composite glyph member -->
    <GlyphName>gravecomb.case</GlyphName>
    <GlyphName>gravecomb</GlyphName>
    <GlyphName>grave</GlyphName>
    <!-- Fallback glyphs if gravecomb.case is not found -->
  </Member>
  <AlignHorizontal>
    <!-- Align the two glyph members horizontally -->
    <Source> <!-- Target glyph is a.smcp -->
      <Member>1</Member>
      <Position>Center</Position>
      <!-- Align to its center (spelling must be US -->
    </Source>
    <Dest>
      <!-- Member to be aligned with the target -->
      <Member>2</Member>
      <!-- gravecomb.case or grave diacritic -->
      <Position>Center</Position>
      <!-- Align center of diacritic -->
      <Offset>16.667%</Offset>
      <!-- Offset center by 1/6th of its width -->
    </Dest>
  </AlignHorizontal>
  <AlignVertical> <!-- Position the diacritic vertically -->
    <Source>
      <Member>1</Member>
      <Position>Top</Position> <!-- Align with top of the base glyph -->
      <Offset>22.17%</Offset> <!-- Offset above the base glyph by 22.17% of its height -->
    </Source>
    <Dest>
      <Member>2</Member>
      <Position>Middle</Position>
    </Dest>
  </AlignVertical>
</Composite> <!-- End of composite glyph definition -->

```



The vertical position for diacritics above Capitals, Small Capitals, and Petite Capitals was calculated using a table to maintain the spacing in direct proportion to the glyph height. Units are funits @ 2048 / em.

Metric	Capital	Small Capital	Petite Capital
Cap Height	1493	1248	1120
Accent Line	1824	1525	1368
Spacing	331	277	248
% of Cap Height	22.17	22.20	22.14
Glyph Height	100%	83.59%	75.02%



Inherit Side-bearings

<InheritFromGlyphLSB> and <InheritFromGlyphRSB> can be used within definitions for composite glyphs like ligatures, which need to use the left and right side-bearings from two different glyphs.

```

<GlyphName>ae.smcp</GlyphName>
  <Member id="1">
    <GlyphName>a.smcp</GlyphName>
    <InheritFromGlyphLSB>TRUE</InheritFromGlyphLSB>
  </Member>
  <Member id="2">
    <GlyphName>e.smcp</GlyphName>
    <InheritFromGlyphRSB>TRUE</InheritFromGlyphRSB>
  </Member>
  <AlignHorizontal>
    <Source>
      <Member>1</Member>
      <Position>Right</Position>
    </Source>
    <Dest>
      <Member>2</Member>
      <Position>Center</Position>
    </Dest>
  </AlignHorizontal>
</Composite>

```

OpenType Figures

Some fonts may need several types of figures. The most common default figures, which align with the tops of capitals are called "Lining Figures." OldStyle figures, which align with the x-height are also known as non-lining figures. Again, in most fonts the figures are monospaced, but fonts with proportional default figures may have Tabular Figures for use with tabulated data. Fonts that have default tabular figures, may have "Proportional Figures" for use with text. By using OpenType features one can include different sets. Name the glyphs appropriately, and Complete Composites will do what it can to generate appropriate glyphs and align them. Some adjustments will be needed, but most of the work is done automatically. The definition below will generate an OldStyle zero and scale it to match the x-height: The scale factor of 0.75 won't be correct for all fonts, but in any case this is just the first step in designing the glyphs. Make them simple and edit to suit the font design.

```

<GlyphName>zero.onum</GlyphName>
  <Member id="1">
    <GlyphName>zero</GlyphName>
    <UseMetrics>TRUE</UseMetrics>
    <YScale>0.75</YScale>
  </Member>
</Composite>

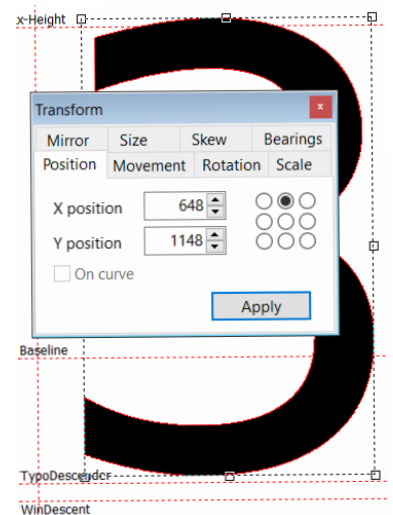
```

The definition below will generate an OldStyle three, and position it just overshooting the x-height: The x-height is 1120 funits and the figure three is 1549 funits high; 1.8% of 1549 is about 28 funits, so the top of the glyph is positioned at 1148 funits. The figure three usually has a rounded contour at the top, whereas figures five and seven are more often flat at the top. They are therefore not given any offset, but are aligned precisely with the x-height. Figures six and eight are identical to the regular figures in most cases, and can be left as composites.

```

<Composite><!-- Non-lining three -->
<GlyphName>three.onum</GlyphName>
  <Member id="1">
    <GlyphName>three</GlyphName>
    <UseMetrics>TRUE</UseMetrics>
  </Member>
  <AlignVertical>
    <Source>
      <Position>xHeight</Position>
    </Source>
    <Dest>
      <Member>1</Member>
      <Position>Top</Position>
      <Offset>-1.8%</Offset>
    </Dest>
  </AlignVertical>
</Composite>

```



Tabular figures would be needed in fonts that have proportional figures by default. No scaling or vertical repositioning of the glyphs will be needed, but the widths need to be uniform. Complete Composites assumes that figure zero will have the greatest advance width, and uses that to calculate the width of tabular figures. Each glyph will need to be adjusted horizontally after generation to centre it optically within the advance width. Transform scripts can be used to centre align glyphs within the advance width.

```
<Composite><!-- Unmapped Tabular one -->
<GlyphName>one.tnum</GlyphName>
<InheritFromGlyphAW>zero</InheritFromGlyphAW> <!-- Use advance width of glyph zero -->
  <Member id="1">
    <GlyphName>one</GlyphName> <!-- Insert glyph one -->
  </Member>
</Composite>
```

It is allowable to mix <GlyphName> with <GlyphMapping> in the same definition but if both are used for the same glyph member the one defined with a mapping will be used first. I use decimal code-points because I find them easier to remember (Hold Alt and type 48 on the numeric keypad to get a zero), but one can also use Hexadecimal code-points (\$30 ≡ 48, *i.e.* 3x16).

One could use, for example:-

```
<Composite><!-- Unmapped Tabular one -->
<GlyphName>one.tnum</GlyphName>
<InheritAW>$30</InheritAW>
  <Member id="1">
    <GlyphMapping>$31</GlyphMapping>
  </Member>
</Composite>
```

However, one cannot use <InheritAW>zero</InheritAW> as FontCreator expects to find a code-point there, so any code after that line in the script will not be executed. Proportional figures simply replicate the default figures. It is up to the designer how much the glyph's side-bearings or glyph contours should be adjusted to suit proportional figures.

```
<!-- ¶¶ Unmapped Proportional Figures -->
<Composite><!-- Unmapped Proportional zero -->
<GlyphName>zero.pnum</GlyphName>
  <Member id="1">
    <GlyphName>zero</GlyphName>
  </Member>
</Composite>
```

Lining figures are defined in the same way. If the font's default figures are OldStyle non-lining figures like those in Georgia, considerable design effort will be needed to create suitable designs for use with capitalised text. They may or may not be monospaced like Tabular Figures.

```
<!-- ¶¶ Unmapped Lining Figures -->
<Composite><!-- Unmapped Lining zero -->
<GlyphName>zero.lnum</GlyphName>
  <Member id="1">
    <GlyphName>zero</GlyphName>
  </Member>
</Composite>
```

<InheritFromGlyphLSB> and <InheritFromGlyphRSB> can be used to get the side bearings from named glyphs, while <InheritFromGlyphSB> will get both. These commands are useful to get bearings from a glyph other than the current glyph member. For example, in this script for Proportional Figures (pnum) it is used to get the left and right bearings from glyph eight to apply to other digits that would have larger side-bearings:-

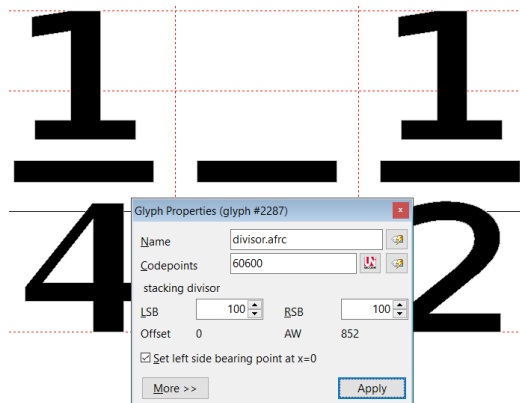
```
<Composite><!-- Unmapped Proportional one -->
<GlyphName>one.pnum</GlyphName>
<InheritFromGlyphLSB>eight</InheritFromGlyphLSB>
<InheritFromGlyphRSB>eight</InheritFromGlyphRSB>
  <Member id="1">
    <GlyphName>one</GlyphName>
  </Member>
</Composite>
```


<InheritFromGlyphAW> gets the width of a named glyph while the short form <InheritAW> can be used instead of the legacy command <InheritAdvanceWidth>.

<InheritFromGlyphAW> is used below to ensure that all single alternative fractions are the same width. If the width is incorrect, edit the divisor.afrc glyph first, then recompose the fractions.

Alternative Fractions

```
<Composite><!-- Alternative Fraction 1/4 -->
<GlyphName>onequarter.afrc</GlyphName>
<InheritFromGlyphAW>divisor.afrc</InheritFromGlyphAW>
  <Member id="1">
    <GlyphName>onenumerator.afrc</GlyphName>
  </Member>
  <Member id="2">
    <GlyphName>divisor.afrc</GlyphName>
  </Member>
  <Member id="3">
    <GlyphName>fourdenominator.afrc</GlyphName>
  </Member>
</Composite>
```



The way that Alternative Fractions are designed is intended to make later editing much easier. Including the numerators and denominators in the exported font is not essential, but if they are deleted the composites that they use will be converted to simple glyphs, so the font will probably get bigger, not smaller. By following the right sequence a lot of unnecessary editing can be avoided:-

1. Run the transform script to insert the glyphs, and let Complete Composites generate the numerators and denominators by scaling the default figures in the font.
2. Use the preview toolbar to see how the fractions look when mixed with default figures. Are they too heavy or too light? Is the spacing between them correct?
3. Edit the single numerators and denominators, and the divisor carefully to remove any intersecting contours, smooth the outlines, and adjust the weight as required. The script can also be edited to adjust the bold effect, and run again after deleting the glyphs created by the first attempt.
4. Now edit the double numerators from 11-63, and the denominators 10, 16, 32, and 64 to adjust alignment and spacing. The numerators get their advanced width from the denominators, so edit the denominators first.
5. Make the numerators and denominators simple to avoid composites of composites.
6. Add an OpenType Alternative Fraction feature to make use of these new glyphs.
7. Add kerning pairs for 1/16, 1/32, 1/64 etc, with the straight quotes " and ' for inches and feet.

Definitions to Aid Design

FontCreator 12.0 added a new feature to create diacritical characters using anchors. This often works better than Complete Composites to position the diacritics correctly, especially with italic typefaces. However, Complete Composites is still very useful for using with transform scripts, e.g. to insert a complete set of Petite Capitals or Small Capitals. If fonts contain the basic Greek or Cyrillic glyphs, there are also scripts for inserting Petite or Small Capitals for Greek or Cyrillic.

Another area where Complete Composites is useful is in generating symbols. By using it to create a minus sign from the plus sign, for example, you can ensure that both glyphs have the same advance width, the same weight, and the same vertical position for alignment with figures in formulae like $3+4-2=5$. Likewise, \pm is generated from plus and minus. After using complete composites to generate minus, decompose the composite glyph, switch to point mode, and delete the vertical stroke. For multiply (\times), delete the vertical stroke, duplicate the horizontal stroke, rotate the copy by 90° , get union of contours, and rotate the now symmetrical glyph by 45° about its centre. The advance width and vertical position will match perfectly with the plus sign: $(4 \times 3) \div 6 + 5 = 7$. The divide sign (\div) is composed from minus and the colon. Many more Mathematical Operators like $\approx \leq \geq \equiv$ are generated in the same way. Superset is generated from subset, union from intersection, double and triple integral from integral, etc. If you later wish to refine the design of these composite glyphs, edit their source, and the changes will be reflected in the composed glyph. You may have to regenerate the composite to update the advance width or reposition them to allow for overshoots.

Enclosed Alphanumerics like circled A are generated from the Large Circle in Geometric Shapes, and Superscript glyphs. If the font has no superscripts for capitals, Petite Capitals are Used, if there are no Petite Capitals either, Capitals are used.

Please post details of any errors on the » [High-logic Bug Report Forum](#) «