URW++

Dutch Type Library

DTL OTMaster

's-Hertogenbosch / Hamburg
2013

Typography means more than bringing order to the passing on of information; it means elevating to the sublime the mould in which the process of passing on is cast.

Frank E. Blokland

Limited user rights

You may never use the software to edit data, including but not limited to fonts of which you do not own the rights, including but not limited to intellectual property rights, copyright and rights to trademark, unless the rightful claimant has given his written and signed consent.

Introduction 7
About This Manual 8
First Steps into OTMaster 9
File Menu 9
- **Open** 9
- **Close** 10
- **Close All** 10
- **Save** 10
- **Save As…** 10
- **Save All** 10
- **Import…** 10
  - — **URW++ Character Layout file** [.cha] 10
  - — **Adobe FDK Feature file** [.fea] 10
  - — **Unicode Variation Sequence file** [.uvs] 10
- **Export…** 11
  - — **URW++ Font file** [.ufm] 11
  - — **URW++ Font Metadata file** [.ufm] 11
  - — **URW++ Character Layout file** [.cha] 11
  - — **Adobe Font Metrics file** from 'kern' table [.afm] 11
  - — **Adobe FDK Feature file** [.fea] 12
- **Print…** 12
- **Preferences** 12
- **Quit** 12
- **Messages** 12

Inspecting Tables 13
Editing Tables 17
Edit Menu 17
- **Cut** [and **Delete**] 17
- **Copy** 17
- **Paste** 17
- **Grow** 17
- **Sort** 17
- **Fixup** 17

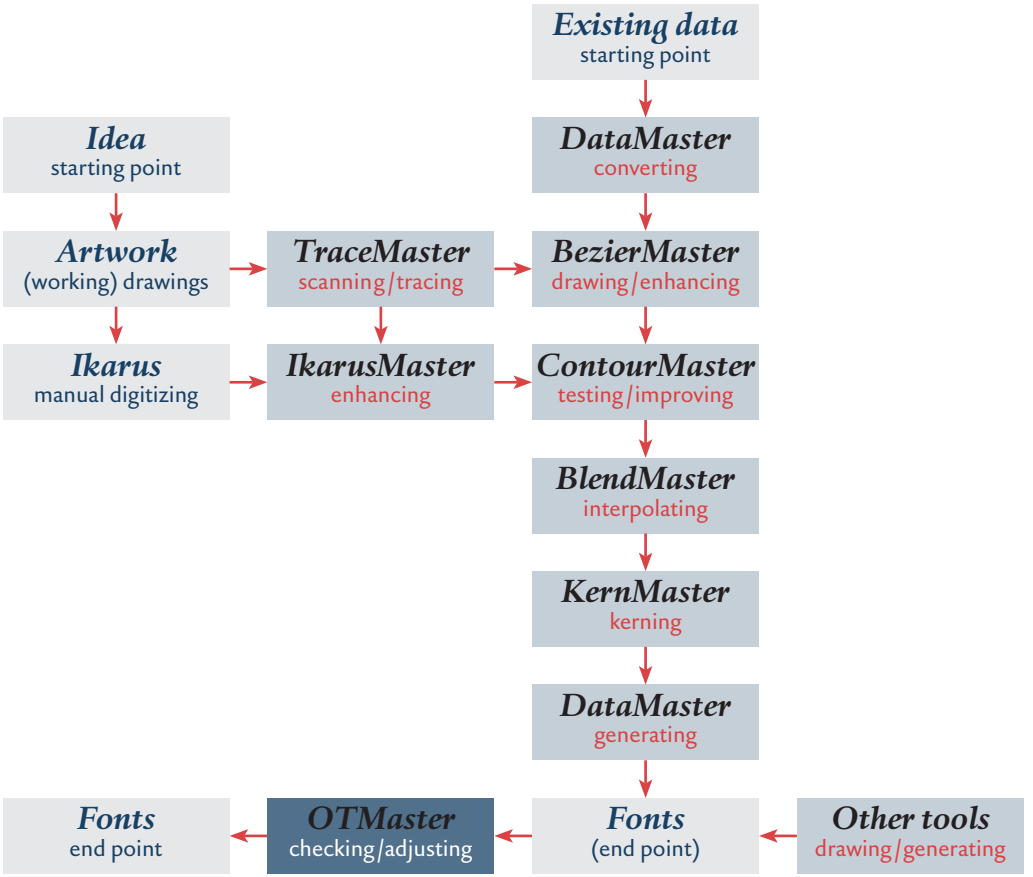
Viewing Tables 18
View Menu 18
- **Nested Tables** 18
- **Text Dump** 19
- **hex** 20
- **dec** 20
- **best fitting** 20

OpenType Font Tables 21
Selected Notes about OpenType Font Tables 22
- — AAT Tables 22
- — CFF 23
- — cmap 26

— COLR, CPAL 28
— DSIG 29
— gasp 29
— glyf, loca, cvt, prep, fpgm 30
— GDEF 31
— GPOS, GSUB 32
— head 35
— hhea 35
— hmtx 36
— kern 37
— maxp 38
— name 39
— OS/2 42
— post 44
TTC Fonts 45
OTMaster's Toolbox 46
Tools Menu 46
Font Viewer 46
Glyph Viewer 48
Side by Side Viewer 50
Embedded Bitmap Viewer 51
Color Viewer 52
'kern' Table Viewer 56
'GPOS'/'GSUB' Viewer 58
Consistency Checker 69
— Header 69
— Name 72
— Version 72
— Statistics 73
— Unicode Ranges 73
— Codepage Ranges 74
— Languages 76
Table Comparator 78
Glyph Copy Tool 79
Glyph Editor 81
File 82
Import… 82
Export… 82
Print… 82
Save… 82
Character Preferences 82
Messages 82
Edit / Selection 82
Revert to Saved 82
Undo 83

Redo *83*
Delete *83*
Cut *83*
Copy *83*
Paste *83*
Paste & Shift *83*
Previous / Next *83*
New *83*
Select All / Deselect All *83*
Select Points *83*
Select Contours *83*
Select Contour Groups *84*
Select Character *84*
Swap Background *84*
View *84*
Tool Bars *84*
Dock Widgets *84*
Display Options *84*
Glyph Set *84*
Reset *84*
Tools *85*
Zoom *85*
Scroll by Hand *85*
Measure *85*
Grid *86*
Guidelines *86*
Scale *86*
Rotate *87*
Affine Transformation *87*
Italization *87*
Mirror and Fold *88*
Hidden Lines *88*
Contouring *89*
T-Disconnector *89*
X-Disconnector *89*
I-Disconnector *90*
Improve *90*
Sense of Rotation *90*
Sequence of Points and Contours *91*
Merge *91*
Character Hinting *92*
Review Changes *92*
Quick Mode *92*
Digitize *93*
Shift *93*

Shift Smooth *93*
Background Glyph *94*
Closing the Glyph Editor *95*
Preferences *96*
— **File Options** *96*
— **User Interface** *96*
— **Foreground Character** *97*
— **Background Character** *97*
— **Autohinting** *98*
— **Shortcuts** *99*
Function and Shortcut Listing *100*



Introduction

DTL OTMaster is a stand-alone application whose interface makes it easy to review and edit .otf and .ttf fonts' tables, no matter whether these fonts have been generated with IkarusMaster, BezierMaster, DataMaster, or any other font editor.

Font editors, like the FontMaster suite, rely on their own internal data formats for type design and font production. With FontMaster, this is either an IK or a BE file, for Ikarus and Bezier outlines respectively, along with various data files for naming font and glyphs, for kerning and definition of typographic layout features. From these data, ready-to-use binary fonts are compiled as the very last step.

OTMaster is a tool whose purpose is to inspect and adjust such ready-to-use binary fonts, irrespective with which font editor they have been created. Its advantage is that it allows editing of tables in a graphic user interface. Moreover, it comes with additional tools like a Glyph Editor to proof, edit and even draw glyph outlines, a 'kern' Table Viewer to proof and

refine the kern table, and a 'GSUB'/'GPOS' Viewer to visually test (and in case of GPOS adjust) these OpenType layout tables.

OTMaster can open, edit and save fonts with SFNT file structure: CFF-based and TT-based OpenType fonts, TrueType fonts and TTC (TrueType Collection) fonts.

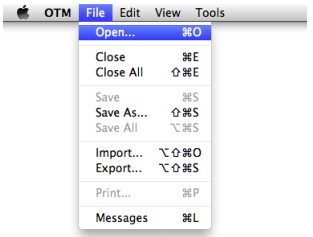
Because OTMaster allows you to edit a binary font's tables, which can be compared to open-heart surgery, it is highly recommended that you know the OpenType specification, at least as regards the tables whose entries you plan to adjust. ►*local* ►*www* (The current online version is 1.6. Please note that as of version 1.5, the OS/2 table has been updated to version 4, and a few nameIDs have been added to the name table.)

About This Manual

There are a number of links in this document. The first type of link is '►*local*' and links to the OpenType specification which Microsoft kindly provided so as to accompany this manual. As long as the folder 'OTM Manual resources' is located next to the manual's PDF file, clicking on such a link will open the according local html page in your web browser, offline. The second type of link is '►*www*' and will open the original online resource. This may be the preferred choice because these data may be more up-to-date. And finally there are internal links like '►*Chapter*', leading from one chapter to another one.

First Steps into OTMaster

When launching OTMaster, the main dialog is empty. Since OTMaster is meant for editing existing fonts, the first step is to open a font.



Drag & drop a font onto OTMaster's main window to open it.

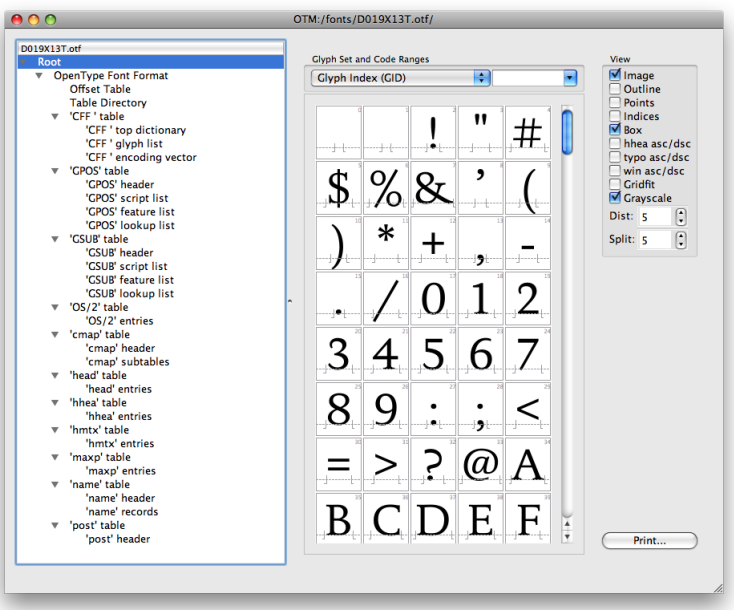
FILE MENU

Open (⌘+O) (CTRL+O)

This opens an existing font using the standard Open dialog.

Alternatively, drag & drop a font file's icon onto OTMaster's main dialog. Or drag & drop one or more font files onto the OTMaster icon to launch OTMaster and open the fonts.

Tip: It is highly recommended that you only edit copies of fonts with OTMaster, to prevent that you inadvertedly overwrite a font when using Save while editing. Either, make a copy of the font file, and edit the copy in OTMaster. Or, Save As… a font immediately after opening it, by another name.



Opening a font will show the table overview to the left, and the content of a selected table to the right.

Close (⌘+W) (CTRL+W)
Closes the currently active font. If you have made any changes which have not been saved yet, a dialog will ask whether you want to save changes now.

Close All (⌘+E) (CTRL+E)
Closes all open fonts. If you have made any changes to a font which has not been saved yet, you will be asked whether to save recent changes or not.

Save (⌘+S) (CTRL+S)
Saves the currently active font at its current location and thus overwrites the existing file.

Save As… (⌘+⇧+S) (CTRL+⇧+S)
Saves the currently active font but allows you to determine a new location in the standard Save dialog.

Save All (⌘+L) (CTRL+L)
All open fonts will be saved at their current locations.

Import…
This dialog's options popup currently offers the following file types:

— URW++ Character Layout file [.cha]
Importing a text-based .cha file will allow OTMaster to show this file's glyph names in tables' **Comment** column. This may be helpful if you have generated a CID-keyed font from FontMaster and plan to edit it in OTMaster. Note that importing a .cha file does not have any impact on the font file or any of its tables!

OTMaster accepts .cha files that map, per glyph, its glyph index, Unicode codepoint and glyph name. Please see the column to the right.

— Adobe FDK Feature file [.fea]
This will import – i.e. compile – selected OpenType layout tables from a .fea file which conforms to Adobe's feature file syntax as described in the *OpenType Feature File Specification.* ▶ *www*

In the dialog which opens next, you may define which of the tables BASE, GDEF, GPOS, GSUB and name you would like to compile from the .fea file's data.

OTMaster does not return any detailed report in case that import, i.e. compilation, fails because of mistakes in your .fea file! So please make sure that your .fea file does not contain any syntax or other errors.

— Unicode Variation Sequence file [.uvs]
Unicode Variation Sequences can be imported from a text-based .uvs file. Please see the ▶ *cmap* chapter.

```
Version 002.000
Starttable
GlyInd;UNINum;PSName
0;;.notdef
1;x0020;space
2;x0021;exclam
3;x0022;quotedbl
4;x0023;numbersign

Endtable
```

*A .cha file as exported by OTMaster and accepted for import. The header must read '**Version 002.00**'. Between tags '**Starttable**' and '**Endtable**', a semicolon-separated table holds the glyph information. The table's header indicates that each glyph is identified by glyph index (**GlyInd**) and optionally is provided with a Unicode codepoint (**UNINum**) and a glyph name (**PSName**).*

The order of columns is arbitrary, but it is important that the order of data (in each row) matches the order determined in the table header.

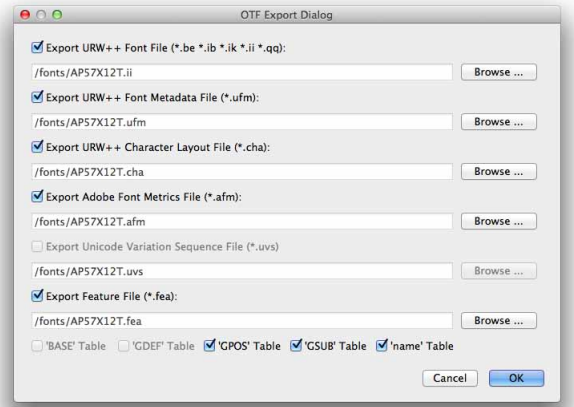
*OTMaster .cha files differ from those employed in FontMaster. OTMaster identifies glyphs by glyph index (**GlyInd**) while FontMaster identifies glyphs by a URW number (**URWNum**). To exchange .cha files between OTMaster and FontMaster, just replace the keyword '**GlyInd**' by the keyword '**URWNum**' (or the other way round). Please note that FontMaster .cha files may hold further information in additional columns. Keywords are described in the FontMaster manual.*

*Being semicolon-separated, a .cha file can easily be edited e.g. in Excel: Add the suffix '**.csv**' to the .cha file's file name and in Excel's Open dialog select 'Text (*.prn; *.txt; *.csv)' as file type. After adjusting the table and saving it, remove the '**.csv**' suffix from the file name.*

***Tip:** Remember that you can define* name *table records in feature file syntax too.*

Export…

Currently, these data may be exported:



— URW++ Font file [.be .ib .ik .ii .qq]

Depending on the input font format, one of the following Ikarus and FontMaster related formats will be generated for reuse in FontMaster: Cubic Bezier data, Ikarus data or Quadratic Bezier data, plus variants of the former two which may include instructions for intelligent scaling (commonly called 'hinting').

— URW++ Font Metadata file [.ufm]

A .ufm file contains the font metadata information which FontMaster needs for generating fonts, including names and vertical metrics. Detailed information can be found in the FontMaster manual.

— URW++ Character Layout file [.cha]

A .cha file as exported by OTMaster maps, per glyph, its glyph index, Unicode codepoint and glyph name.

An OpenType font's .cha file is helpful when importing this font in FontMaster: if you replace 'GlyInd' by 'URWNum' in the .cha file's header, the original glyph indices will serve as URW numbers which identify glyphs in the FontMaster suite.

— Adobe Font Metrics file from 'kern' table [.afm]

This will export an .afm file with the kern table's kerning – if there is a kern table in the font.

An .afm file contains font metrics and basic font metadata. For details see Adobe's *Font Metrics File Format Specification*. ►*www*

Note: Feature files usually identify glyphs by their glyph names. This however means that if you 'transfer' typographic layout features from one of your fonts to another one, glyphs in both fonts need to share identical names, and all glyphs referenced in the feature file must be present in the destination font.

*Please mind that such a 'transfer' is not lossless because it involves interpretation: 1. **Export** will dump OpenType layout tables into an AFDKO-syntax feature file, and 2. **Import** will compile OpenType layout tables from a feature file.*

Also, 'transferring' features from one font to another often is not that rewarding because most likely each font has a different glyph set with different alternates, so that different feature behavior is required.

Note: Since an .afm file's metrics are supposed to relate to an implicit UPM = 1000, OTMaster will 'scale' metrics in case if the font's UPM is not 1000.

— Adobe FDK feature file [.fea]

This will export selected OpenType layout tables as a .fea file in Adobe's feature file syntax. As with the import function, you may choose from BASE, GDEF, GPOS, GSUB and name. Export means that the tables will be dumped which involves interpretation as described in the note on the previous page.

Please consult Adobe's *OpenType Feature File Specification* for details about the feature file keywords, syntax and examples. ▶ *www*

Print … (⌘+P) (CTRL+P)

Because it does not make much sense to print data as represented in the user interface, it is recommended that you switch, in the **View** menu, to **Text Dump** mode and save either the text dump or XML files and print this.

Preferences (⌘+,) (CTRL+,)

Please see the ▶ *Preferences* chapter.

Note: In the Mac OS version of OTMaster, both ***Preferences*** *and* ***Quit*** *are located in the* ***OTM*** *menu.*

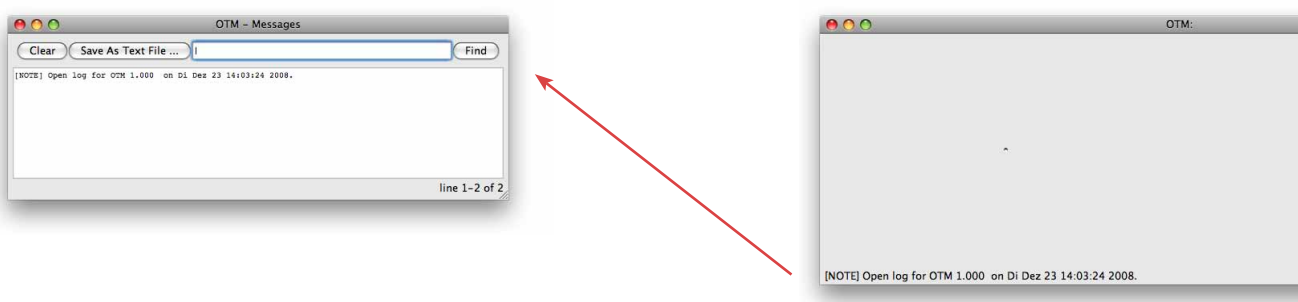
Quit (⌘+Q) (CTRL+Q)

This will quit the application. If fonts have been changed but not saved yet, you will be asked whether you want to save these changes or not.

Messages

This opens the Messages window which collects all status messages as shown at the bottom of the OTMaster's main dialog.

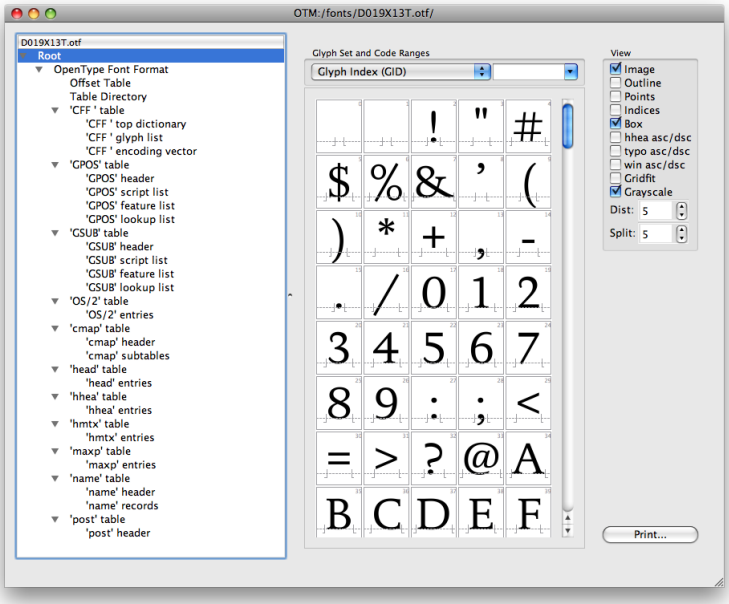
You may **Clear** the panel, **Save As Text File …** its content, and search in it with help of **Find**.



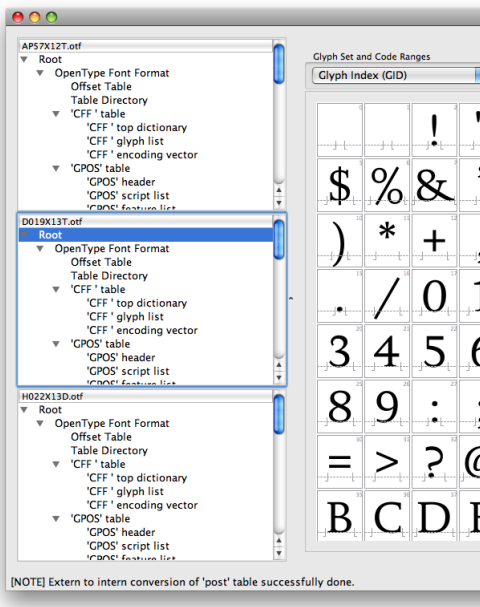
A typical status message. All of them will be collected in the Messages window.

Inspecting Tables

Once a font is opened, OTMaster's main dialog consists of two areas:



OTMaster's main window showing a CFF-based OpenType font. If more than one font is opened, the table overview is tiled into as many segments as there are open fonts. The currently active font's segment 'glows' in the Mac OS version:



The leftside area presents a table overview with the font's file name in a header. If more than one font has been opened, this area consists of as many segments as there are open fonts. The rightside area shows the entries of the table which is currently selected in a font's table overview.

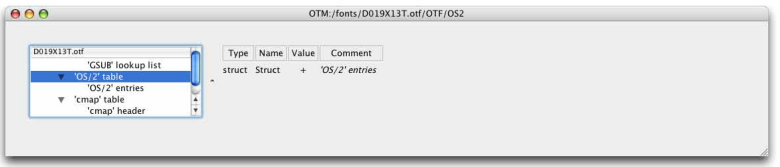
The table overview usually starts with the Root entry which represents the font as a whole. Root is the default selection in the table overview, and accordingly the rightside content area presents this font's glyph set. Strictly speaking, this glyph set overview is merely for your convenience – Root does not represent any specific table. Functionality and options for Root glyph overview and Font Viewer are identical and are described in the chapter ▶ *Font Viewer*.

Root includes OpenType Font Format which indicates that a font is an OpenType font. This in turn is populated with the OpenType font's data according to SFNT file structure: An Offset Table tells how many tables this font contains, plus information for binary search. The Table Directory points out start (offset) and length of every table. These then are followed by all tables present in the font. (A TTC font would indicate TrueType Collection Format rather than OpenType Font Format. The overall structure would differ slightly too. Please consult the chapter ▶ *TTC Fonts*.)

Offset Table and Table Directory are not supposed to be adjusted manually. They are read-only and will be updated automatically as soon as tables are modified, removed or added.

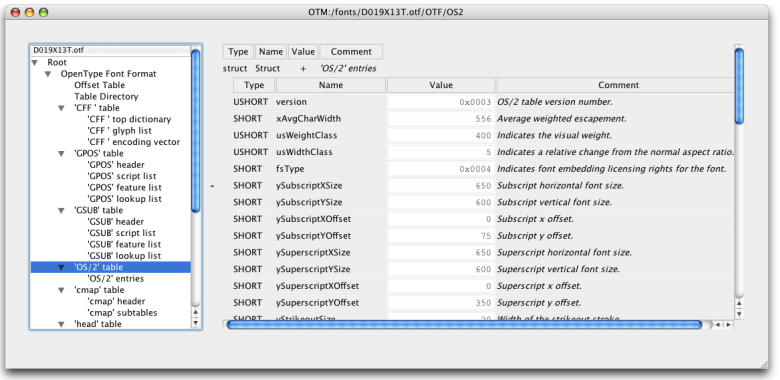
The term OpenType Font Format refers to both CFF-based and TT-based OpenType fonts. The former store outline data in a CFF table, the latter store them in a glyf table.

Simple tables include only [table] entries. Selecting [table] table in the table overview will exhibit the top level of this table in the content area.



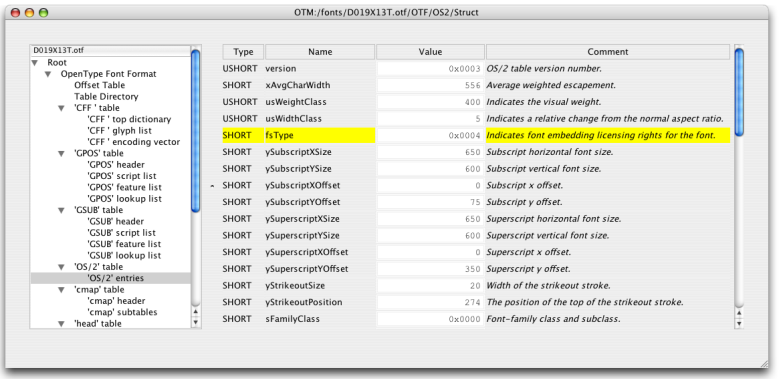
A table's top level.

The **+** sign in the **Value** column indicates that there is more information, and a click on the **+** will fold out a nested table whose content is indented:



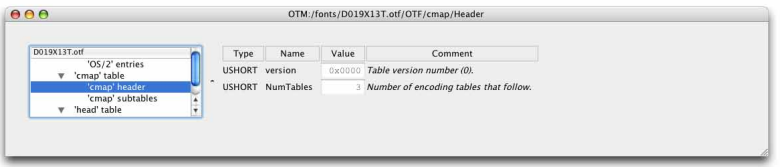
Accessing a table's entries via its top level – by unfolding a nested table.

Since the top level of a table is not very informative, most of the time you may want to select [table] entries to access the table's entries directly, ready to be studied or adjusted:



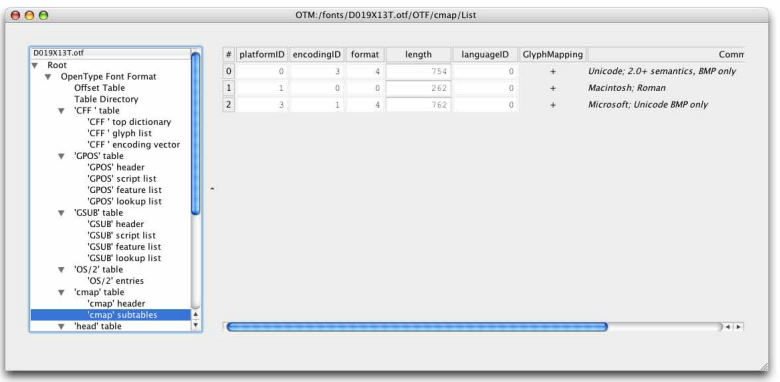
Accessing a table's entries directly.

Tables of variable length which consists of one or more subtables like cmap or kern, or multiple entries like name, then [table] table will include both [table] header and [table] entries. Like Offset Table and Table Directory, the [table] header is read-only:



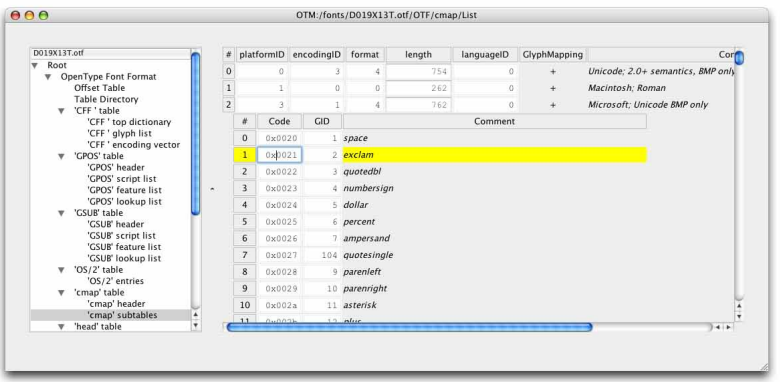
Header of a table which contains multiple subtables.

The cmap table is a good example for a table which consists of multiple subtables. This is reflected in the 'cmap' entries table content area:



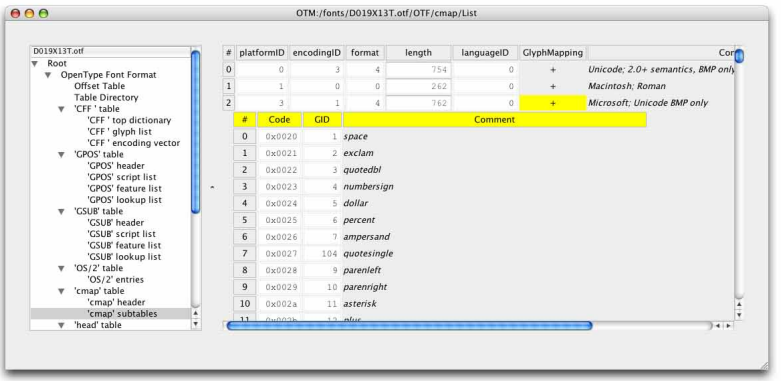
A table's 'table' entries shows the available subtables in the content area, ready to be unfolded.

And again the **+** (in the **GlyphMapping** column) signals that per subtable there is some content to be unfolded. Clicking on one of them will reveal the respective subtable's mappings of Unicode codepoints (**Code**) to glyph indices (**GID**). In our example, we click the last subtable's **+** to see it's entries. (Clicking the **+** again will hide it.)



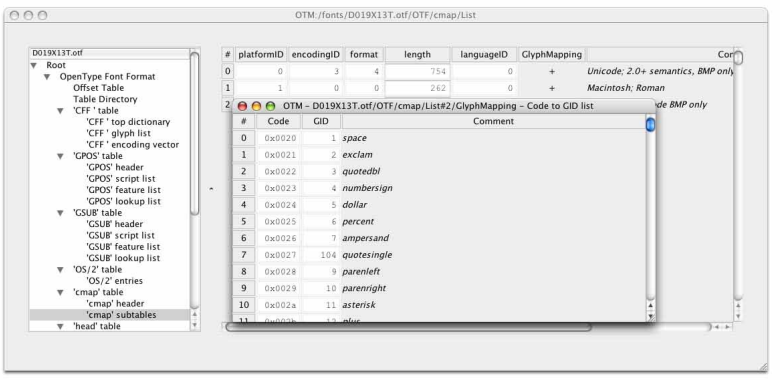
*Unfold a subtable by clicking the **+** sign and review or edit entries. The subtable's entries are indented.*

However, some tables like GSUB or GPOS have a rather complex structure, which makes it hard to keep track of the hierarchy and the level to which displayed entries belong. To address this, and to help comparing tables of different fonts, OTMaster can display a table's or nested table's content in a separate window. Just double-click onto a nested table's header, like the one selected (and thus colored yellow) in the image below



Click onto a nested table's header …

to create a new window. This way, you can compare multiple font's tables, and as many as you like. The only limitation is the size of your screen!



… to open a new window for this nested table.

We have already seen some important OTMaster actions:

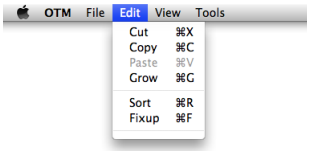
1. Click a **+** sign to show or hide a nested table.
2. Change a table's entries in usual textboxes.
3. Double-click a nested table's header to show its content in a new window.

And one addition:

4. Switch from column to column with ⇥ (to right) and ⇧+⇥ (or ⇤; to left) and from line to line with ↑ (arrow up) and ↓ (arrow down) keys.

Editing Tables

The most important functions for removing or adding data are found in the **Edit** menu. All of these functions apply to individual tables selected in the table overview, or table entries selected in the table content area.



EDIT MENU

Cut [and **Delete**] **(⌘+X) (CTRL+X)**
Cuts a selected table or table entry. This will remove the table from the font, or entry from the table, and keep it in the clipboard – this means, Cut is equivalent to **Delete**.

*Note: There is no special **Delete** function in OTMaster. However, **Cut** serves the same purpose.*

Copy (⌘+C) (CTRL+C)
Copies a selected table or table entry into the clipboard without removing it from the font.

Paste (⌘+V) (CTRL+V)
Pastes a table from the clipboard into the currently selected font, or pastes a table entry from the clipboard into the currently selected table.

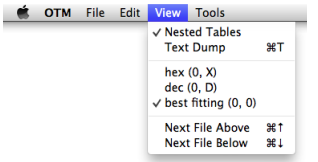
Grow (⌘+G) (CTRL+G)
This function does exactly what it says – help growing a table. In tables of variable length (such as name, cmap, kern) select an entry (like a name record in the name table) and use Grow to duplicate it. In other tables, Grow will create a new, empty entry.

Sort (⌘+R) (CTRL+R)
Sorts table entries in tables of variable length. For example, this will sort name table entries by platformID – encodingID – languageID – nameID, or sort cmap table entires by Unicode codepoints.

Fixup (⌘+F) (CTRL+F)
Use this to remove duplicate table entries from tables of variable length (like name, cmap, kern). This is helpful, for example, if you have used **Grow** to duplicate an existing name table name record but then find that you do not need an additional name record.

Viewing Tables

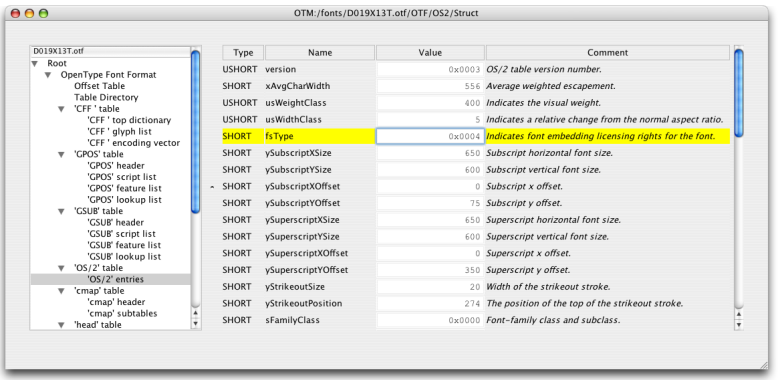
OTMaster offers two modes for viewing tables and two modes of table value representation representing table values.



VIEW MENU

Nested Tables (⌘+N) (CTRL+N)

This mode presents each table's content as a typical dialog. With tables like head, hhea, OS/2 there are, per entry, its data **Type**, its **Name** and finally the **Value** in a textbox which may be edited in most cases. Where possible, there is an additional **Comment** column with a brief description. This way, OTMaster is not just a table editor but at the same time provides a built-in documentation for most of the table data. For example, OTMaster exposes glyph names in the **Comment** column which is particularly useful since tables identify glyphs by mere glyph indices.



The Nested Tables mode. Edit table entries in usual textboxes!

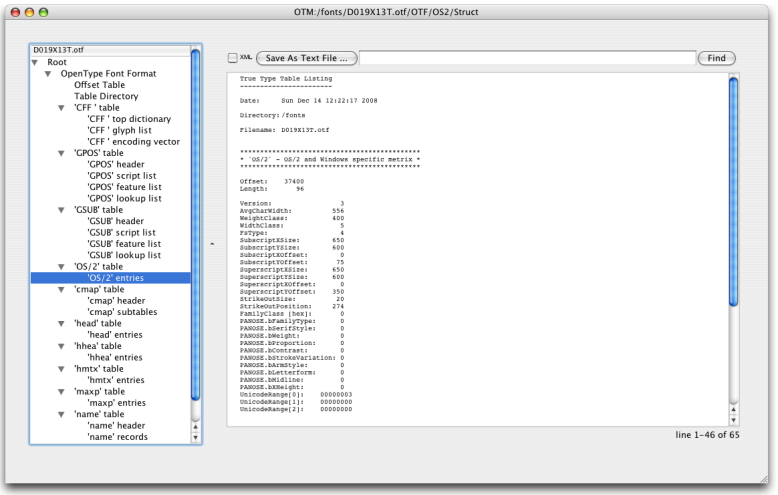
The Nested Tables mode exposes only the top level, indicating the existence of a nested table by a **+** sign which, when clicked, will show or hide a nested table's entries.

Double-click a (nested) table's header to display its content in a separate window.

While in Nested Tables mode, a right-click into the content area will open a context menu which offers **Text Dump** and **XML Dump** modes (see the next page for details). Choosing either of them will open a new window.

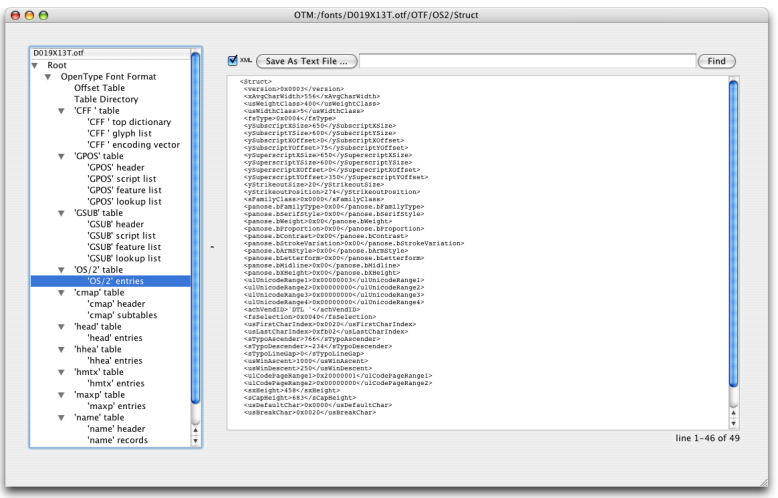
Text Dump (⌘+T) (CTRL+T)

Text Dump will present table data as text. Please note that with complex tables – like glyf, CFF, GSUB or GPOS – this text is rather a summary:



A table's content in Text Dump mode.

While in Text Dump mode, the **XML** checkbox at the top of the table content area allows you to switch to XML mode which will format the summary in XML style:



Text dump in XML style. Please note that this is not compatible with TTX's XML-style dump!

Save As Text File ... will save a plain or XML text dump – useful for comparison in a text editor, or making printouts for proofreading.
Find will search the (XML) text dump for a string.

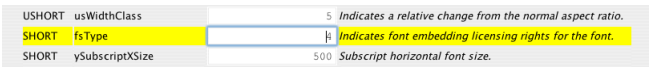
hex (⌘+H) (CTRL+H)

Hex shows values as hexadecimal numbers:

This example is taken from the OS/2 table entries.

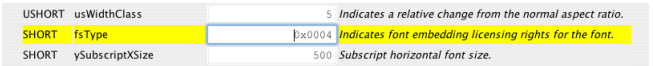
dec (⌘+D) (CTRL+D)

Dec shows values as decimal numbers:



best fitting (⌘+B) (CTRL+B)

Best fitting will choose the most appropriate representation of values, which is hexadecimal or decimal. This is the default.



*You will notice that with **best fitting**, which is the default, some values are decimal while others are hexadecimal – this choice is built into OTMaster.*

OpenType Font Tables

You are expected to know the OpenType specification, rudimentarily at least, and at least as regards the tables whose data you intend to adjust. It is not the task of this manual to reproduce or rephrase them here, though this cannot be avoided at times. ▶*local* ▶*www* What you will find in this chapter though is a brief summary of an OpenType font's structure, a list of common tables, with a few additional notes about things to consider: first, to make sure that data across tables is consistent, and second, to provide some OTMaster tricks for adjusting and extending tables.

As indicated in the ▶*Inspecting Tables* chapter, the structure of an OpenType font is quite simple. It starts with header information. First, an Offset Table whose SFNT version indicates whether the font is TT-based (0x001000), CFF-based (string 'OTTO') or a TrueType Collection (string 'ttcf'), and the number of tables in case of a TT/CFF-based OpenType font. Second, a Table Directory which points out where each table starts and how long it is. Header information are directly followed by the tables.

The OpenType specification provide the following list of tables:

1. Required tables
   - cmap character to glyph mapping
   - head font header
   - hhea horizontal header
   - hmtx horizontal metrics
   - maxp maximum profile
   - name naming table
   - OS/2 OS/2 and Windows specific metrics
   - post PostScript information

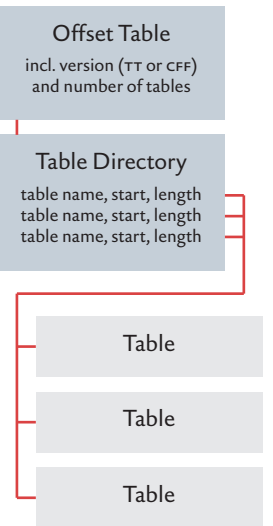
2. Tables in TT-based OpenType fonts
   - cvt control value table
   - fpgm font program
   - glyf glyph data
   - loca index to location*
   - prep CVT program

3. Tables in CFF-based OpenType fonts
   - CFF PostScript font program (Compact Font Format)†
   - VORG vertical origin

4. Tables for bitmaps
   - EBDT embedded bitmap data
   - EBLC embedded bitmap location data
   - EBSC embedded bitmap scaling data

Info: The OpenType specification. ▶local ▶www Especially see The OpenType Font File *▶local ▶www and* Recommendations for OpenType Fonts *▶local ▶www which comes with additional information and clarifications.*

An OpenType font's data structure (simplified):



** Location means: location of glyphs' data inside of the glyf table. Effectively, the loca is the glyf table's 'external' header or directory.*

† The CFF table actually is a font in its own right, holding not only outline data, but also glyph- and font-level hinting information, font names and more.

5. Typographic layout tables
   - BASE baseline data
   - GDEF glyph definition data
   - GPOS glyph positioning data
   - GSUB glyph substitution data
   - JSTF justification data

6. Other tables
   - DSIG digital signature
   - gasp grid-fitting and scan-conversion procedure
   - hdmx horizontal device metrics
   - kern kerning
   - LTSH linear threshold data
   - PCLT PCL5 data
   - VDMX vertical device metrics
   - vhea vertical metrics header
   - vmtx vertical metrics

OpenType fonts' SFNT structure makes it easy to add tables which are not standardized by the OpenType specifications. The only requirement is that their names do not collide with standard table's names.

For example, Microsoft's VOLT – an application for defining typographic layout behavior visually – adds a table called TSIV to the font. This holds the VOLT project data. The TSIV data is removed from the font as soon as you 'ship' a font.

Selected Notes about OpenType Font Tables

Below, you will find some notes about individual OpenType font tables. These are in no way official recommendations but rather point out a few things to consider. You are strongly adviced to compare with the original OpenType specification – links are provided whereever possible.

Tables not yet supported by OTMaster do appear in the table directory and table overview, but no content is shown in the table content area. When saving a font, these unsupported tables will be written back to the font in their original form.

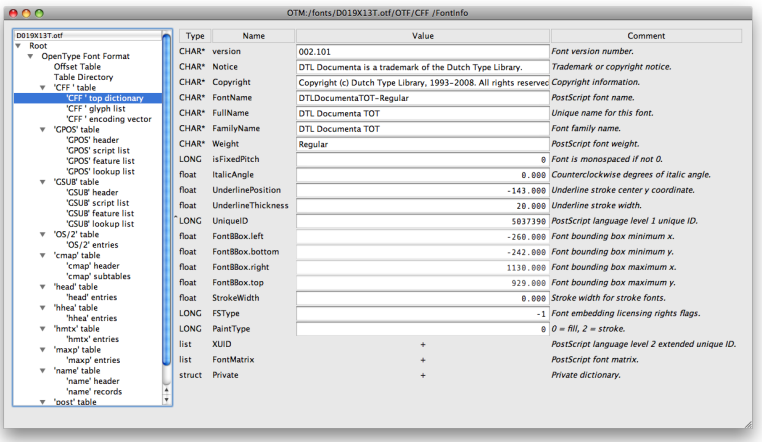
— AAT Tables

AAT (Apple Advanced Typography) tables are not supported.

— CFF

An OpenType font's Compact Font Format table is a complete font. It comes with outline data, but also with font names and (unless CID-keyed) glyph names, font- and glyph-wide hinting instructions, and more.

Info: The CFF *table.* ▶*local* ▶*www Adobe's* The Compact Font Format Specification, #5176. ▶*local* ▶*www Adobe's* The Type 2 Charstring Format #5177. ▶*local* ▶*www*



Note: Adobe discourages the use of XUID *and* UniqueID *in fonts that are not* CJK *fonts.* ▶*www*

The 'CFF' top dictionary. *The* CFF *table's* Name INDEX *shows up as* **FontName** *in this list.*

1. 'CFF' top dictionary contains a CFF font's meta-information:

1.1 CFF and name table: The CFF table's Name INDEX – found in the 'CFF' top dictionary as **FontName** – must be identical to name table entries with nameID 6 (*PostScript* name, both Microsoft platform and Macintosh platform) and nameID 4 (*Full* name, Microsoft platform). If your font is a CFF-based OpenType font and you intend to change these names, then you need to change the 'CFF' top dictionary's **FontName** at first and only then can adjust the name table entries for *PostScript* name and *Full* name.

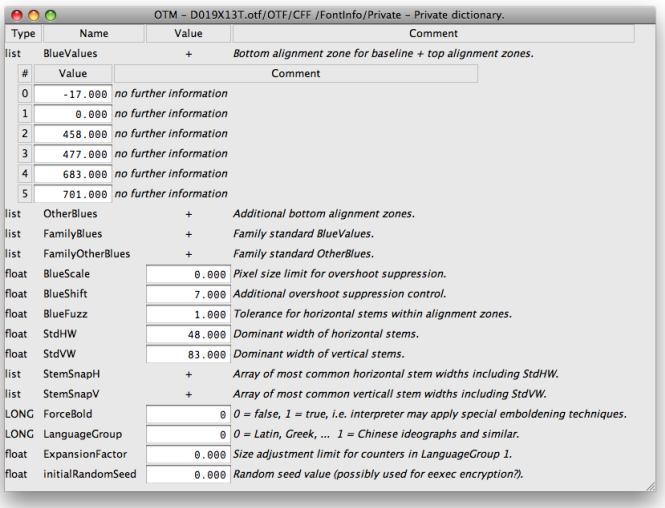
Other font name entries in the 'CFF' top dictionary may reflect the respective name table entries. **Weight** refers to weight only, not to style!

1.2 CFF and post table: Both the CFF and the post table have entries for underline position and thickness. CFF has **UnderlinePosition** and **UnderlineThickness**. post has **underlinePosition** and **underlineThickness**. While both tables' underline thickness entries share the same value, those for underline position differ. The CFF table's **UnderlinePosition** is measured from the vertical center of the underline outline's height, or thickness. The post table' **underlinePosition** is measured from its highest point. Their relation is:

CFF.UnderlinePosition
= post.underlinePosition – (post.underlineThickness / 2)

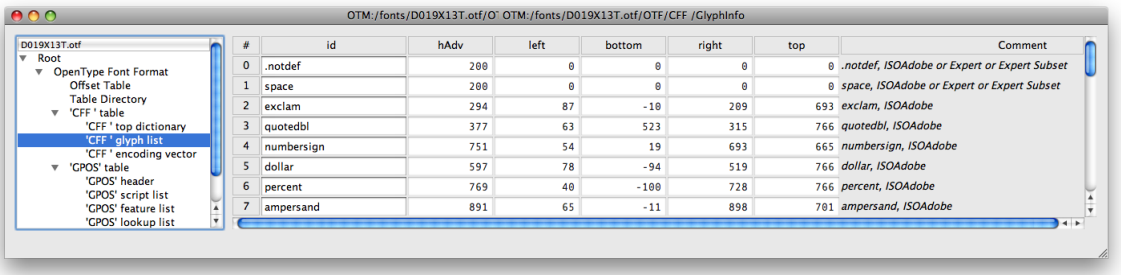
Take care that **ItalicAngle** is identical with the post table's **italicAngle** and matches the hhea table's **caretSlopeRise** and **caretSlopeRun**. More in the ▶*Consistency Checker* chapter.

1.3 Font-wide hinting information –**BlueValues**, **OtherBlues**, etc.– are found in the 'CFF' top dictionary's **Private** nested table:



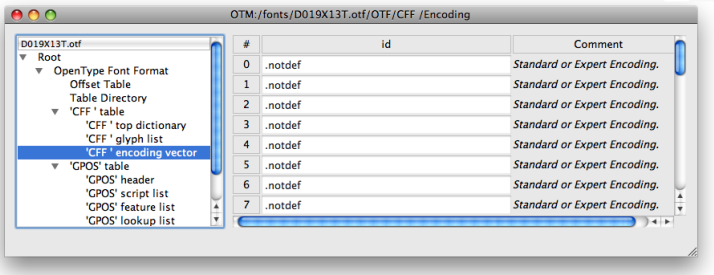
*The **Private** dictionary which is accessed by way of the 'CFF' top dictionary – here displayed in a separate window.*

2. 'CFF' glyph list holds the glyph names as found in the CFF table's CharStrings INDEX. Additional columns present, per glyph,
hAdv, the horizontal advance width,
left, the leftside bearing which is equal to xMin (smallest horizontal extension of the glyph),
bottom, the yMin (smallest vertical extension of the glyph),
right, the xMax (largest horizontal extension of the glyph),
top, the yMax (largest vertical extension of the glyph),
and **Comment** with Adobe's standard glyph names for these glyphs.



The 'CFF' glyph list. This is where you may change glyph names in CFF-based OpenType fonts. in TT-based OpenType fonts, glyph names are found in the post table – unless this is a version 3 post table.

3. 'CFF' encoding vector links any of the characters covered in Adobe's Standard or Expert Encoding to glyph indices. This is read-only.



The 'CFF' encoding vector.

Unlike the glyf table, the CFF table does not show outline coordinates. For editing outlines you may prefer to use the ▶ *Glyph Editor*.

— cmap

Info: The cmap *table.* ►*local* ►*www*

The Character To Glyph Mapping table maps input Unicode codepoints (**Code**) to glyphs which are identified by glyph index (**GID**). It consists of one or more subtables. A standard OpenType font usually contains three of them (platformID – encodingID – format):

0 – 3 – 4 (no platform – Unicode – format 4)
1 – 0 – [6] (Macintosh – Roman – format 6 or other)
3 – 1 – 4 (Microsoft – Unicode – format 4)

The last one is required for Windows. The second one is or may become increasingly obsolete.

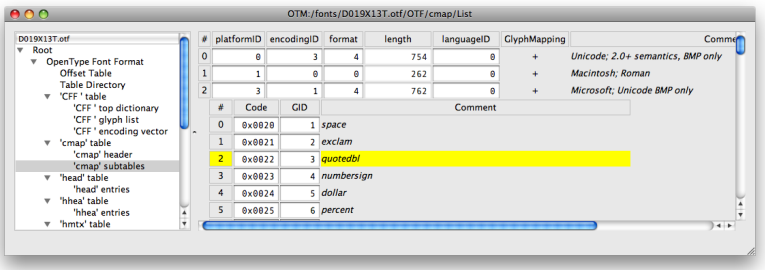
To address supplementary Unicode codepoints beyond BMP, a font also needs a format 12 subtable (platformID – encodingID – format):

3 – 10 – 12 (Microsoft – Unicode UCS-4 – format 12)

Note that this must be present in addition to the format 4 subtable rather than instead of it. This too is required for Windows.

To add mapping information to a cmap subtable, click the **+** to access its entries, select one of them, choose **Edit > Grow** (⌘+G) (CTRL+G) to duplicate the selected entry, and adjust Unicode codepoint and glyph index:

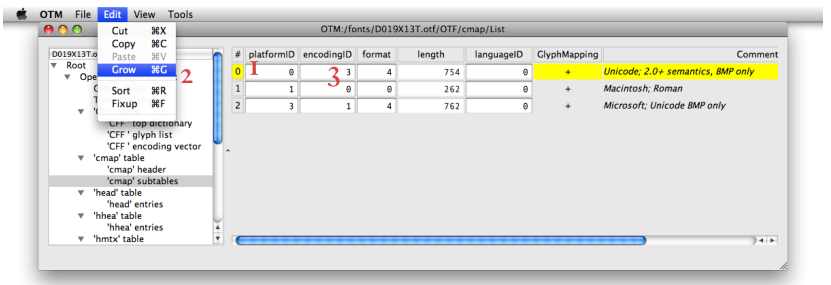
Note: You need to repeat this for every subtable if you want the according mappings to be covered by all of them.



cmap table with three subtables, the last of which is unfolded.

It is possible to create a new subtables of a different format by

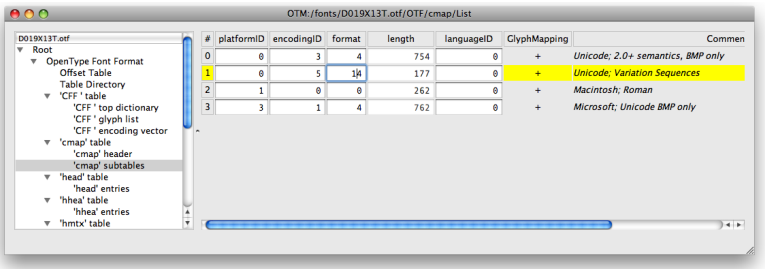
1. selecting an existing subtable,
2. duplicating it with **Grow**,
3. changing the new subtable's format and possibly platformID and encodingID too.



Duplicating a cmap *subtable.*

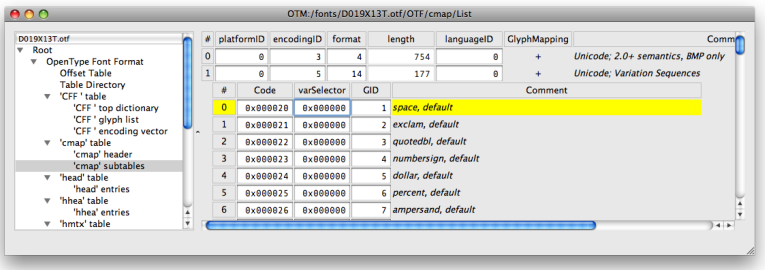
OTMaster supports Unicode Variation Sequences. With the method described above it is possible to add an according subtable. For this, platformID – encodingID – format need to be 0 – 5 – 14.

Note: See the cmap *table specification, 'Format 14: Unicode Variation Sequence'.* ▶ *local* ▶ *www*



Creating a subtable for Unicode Variation Sequences. Change the duplicate subtable's ***format****,* ***platformID*** *and* ***encodingID****.*

A Variation Sequence consists of a codepoint (**Code**) followed by a variation selector (**varSelector**). If this pair is matched in an input string, the glyph mapped to this codepoint will be replaced by another glyph referenced by **GID**. This can be compared to a typographic layout feature, yet it is the input string itself which initiates the replacement by a variant glyph. When creating such a subtable in OTMaster, glyphs' variation selectors are set to zero by default and need to be adjusted manually:



*Adjust the glyph's variation selector (****varSelector****).*

Tip: Use ↑ *(arrow up) and* ↓ *(arrow down) keys to flip through all mapping entries quickly, and* ⇥ *(to right) and* ⇧+⇥ *(or* ⇤*; to left) to jump from column to column.*

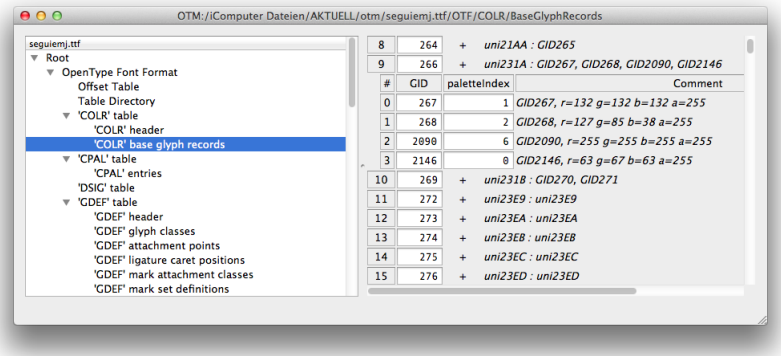
Further information about Ideographic Variation Sequences can be found in Ken Lunde's *IVS Support · The Current Status and the Next Steps.* ▶ *www*
Also see the Unicode Consortium's *Ideographic Variation Database.* ▶ *www*

— COLR

— CPAL

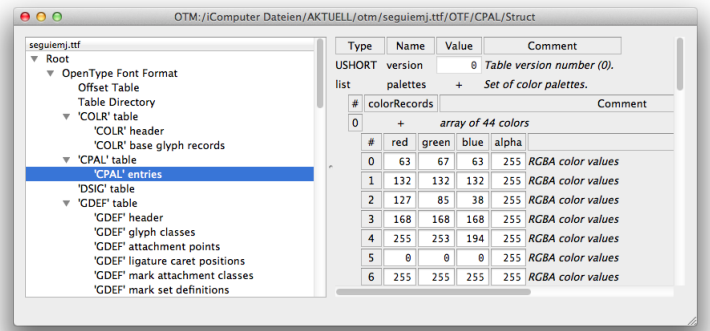
Info: The COLR *table.* ▶*local* ▶*www*
The CPAL *table.* ▶*local* ▶*www*

The COLR table maps (usually encoded) base glyphs to color-specific glyphs and color definitions. The CPAL table holds these color definitions.



The COLR *table with one nested* **GID** *to* ***paletteIndex*** *sublist unfolded.*

The COLR table maps base glyphs, identified by **GID**, to color-specific glyphs, also identified by **GID**, and references to color definitions, by **paletteIndex**.



The CPAL *table with a nested sublist containing the first palette's* ***colorRecords****.*

The CPAL table in turn provides the **colorRecords**. A record's **#** column corresponds to the **paletteIndex** that the COLR table refers to. There may be multiple color **palettes** in the CPAL table, each of which contains a full set of **colorRecords** in a nested list. A color is defined by **red**, **green**, **blue** and **alpha** values.

A special **paletteIndex** value of 0xFFFF indicates that the user-defined foreground color is to be used.

Both tables can be edited the usual way, as described in the previous section about the cmap table.

— DSIG

If the Digital Signature table exists in a font which you are about to edit in OTMaster you may want to delete it with **Edit > Cut** (⌘+X) (CTRL+X). Editing the font would invalidate this table anyway and indicate that this font is not in the state in which it was originally delivered.

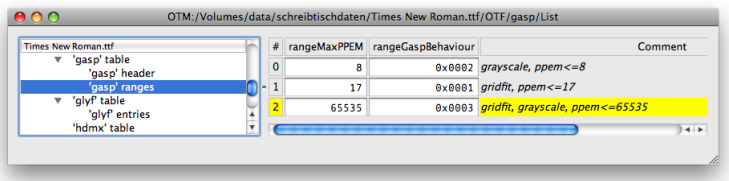
Info: The DSIG *table.* ►*local* ►*www*

Note: It is assumed that you are either editing your own font or have permission to do so!

— gasp

The Grid-Fitting And Scan-Conversion Procedure table determines which kind of rasterization is preferred for specific ppem size ranges.

Info: The gasp *table.* ►*local* ►*www*



Selecting the last record of the gasp *table.*

The upper limit of a range is its maximum ppem size (**rangeGaspPPEM**), the lower limit is 0 or the previous range's maximum ppem size +1. The last record must have a **rangeGaspPPEM** of 0xFFFF or 65 535. Four kinds of preferred rasterization (**rangeGaspBehavior**) have been defined:

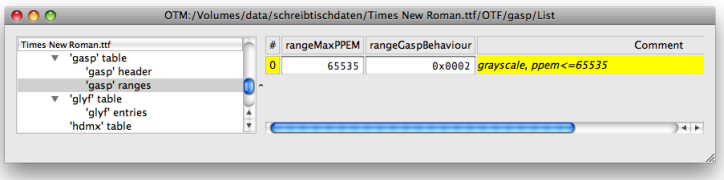
grayscale (0x0002)
gridfit (0x0001)
grayscale & gridfit (0x0003)
none of them (0x0000)

There are three additional values which address ClearType and are considered to be independent of the above ones:

symmetric gritfit (0x0004)
symmetric smoothing (0x0008)
symmetric smoothing & symmetric gritfit (0x000C)

Please note that in the head table, bit 13 of the **flags** entry indicates whether or not a font is optimized for ClearType.

For example, to adjust the above gasp table such that it will encourage grayscaling for all ppem sizes, select the first two records one by one and **Edit > Cut** (⌘+X) (CTRL+X) them. Finally, change **rangeGaspBehavior** to '**0x0002**'.

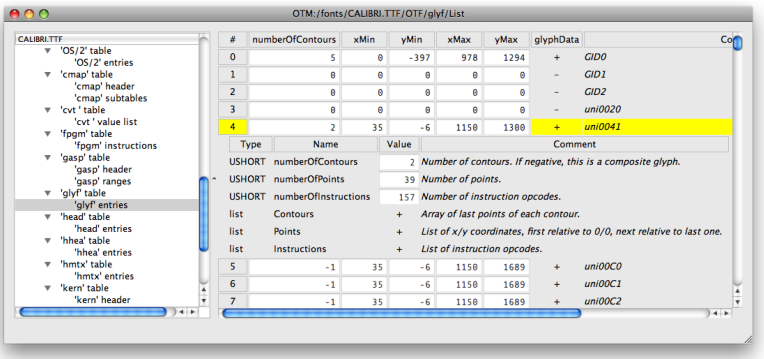


The gasp *table after removing the first two records and adjusting* ***rangeGaspBehavior****.*

— glyf
— loca

The glyf table contains TT outline information as well as glyph-level instructions, i.e. hinting information. This table is read-only in OTMaster, so we do not go into details here. Try the ▶ *Glyph Editor* for editing glyph outlines.

Info: The glyf *table.* ▶ *local* ▶ *www*
The loca *table.* ▶ *local* ▶ *www*



The loca table (not shown here) can be considered as the glyf table's directory or header. Per every glyph index, the loca table points out (by way of an 'offset') where this glyph's data starts in the glyf table.

Just like tables' header information are read-only, the loca table is read-only and will be updated automatically by OTMaster.

— cvt
— prep
— fpgm

The Control Value Table, Control Value Program and Font Program hold font-level instructions. OTMaster indicates their meaning in the **Comment** column.

For details about the above tables and TrueType instructions please see Apple's TrueType specification. ▶ *www*

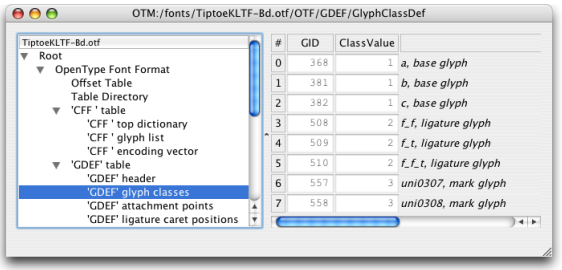
— GDEF

Info: The GDEF *table.* ►*local* ►*www Also see the* OpenType Layout Common Table Formats document. ►*local* ►*www*

The Glyph Definition table belongs to the typographic layout tables and provides additional information to which GSUB and GDEF may refer.

Curently, OTMaster only supports glyph class definitions which are located in 'GDEF' glyph classes. Each glyph, identified by glyph index (**GID**), is associated with no or one of these classes, identified by **ClassValue**:

1. base glyphs
2. ligature glyphs
3. (combining) mark glyphs
4. glyph components

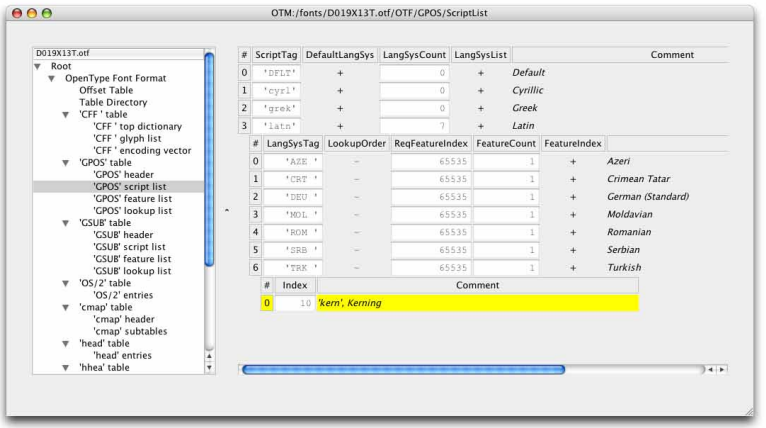


— GPOS

— GSUB

Glyph Positioning and the Glyph Substitution tables start with three lists: script list, language list, and feature list.

Info: The GPOS *table.* ►*local* ►*www The* GSUB *table.* ►*local* ►*www Also see the* OpenType Layout Common Table Formats document. ►*local* ►*www*

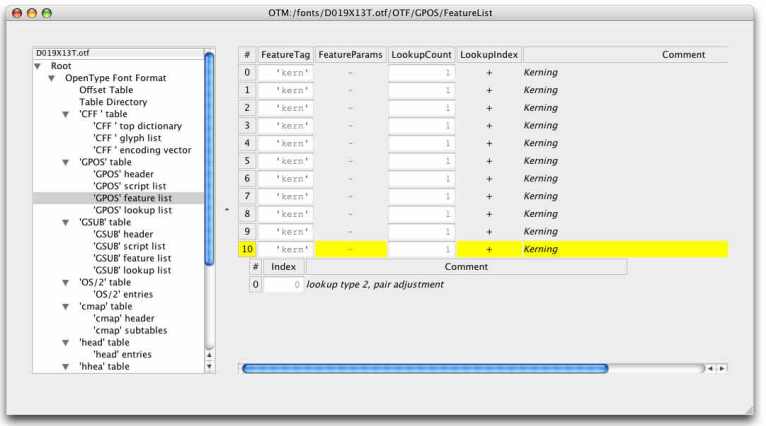


The script list, with unfolded language list for the 'latn' script.

The script list sums up all scripts explicitly addressed by the layout table. These are scripts – or writing systems – like 'latn' for Latin or 'cyrl' for Cyrillic.

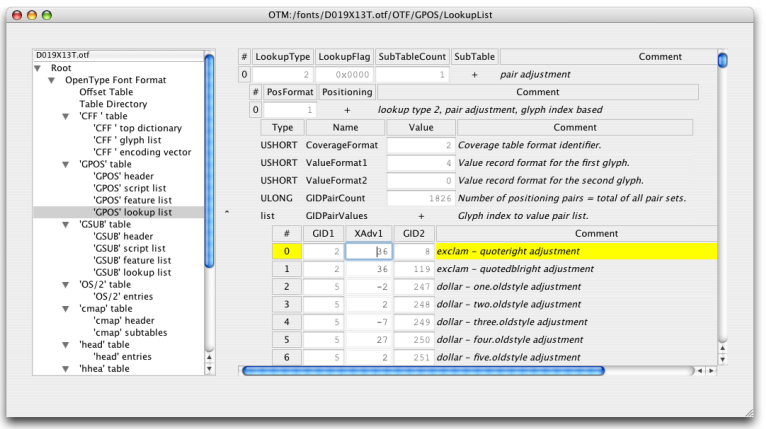
Per each script as found in the **ScriptTag** column, there is a list of languages explicitly addressed by the layout table, plus a 'default' language for which there is a special place in the data structure. (In case that a layout application cannot find a match for the selected language – e.g. by way of the spelling dictionary in Adobe InDesign – in the font's layout table, it would fall back to this 'default' language.) The former is accessed by clicking the **+** in the **LangSysList** column, the latter by clicking the **+** in the **DefaultLangSys** column. In the example above, the 'latn' script's languages are folded out.

Per each language there is a list of features associated with it. This is unfolded by clicking the **+** in the according language's **FeatureIndex** column. In the example above, there is one 'kern' feature associated with the Turkish language or 'TRK' – to know what the actual feature behavior of 'kern' is for Turkish, we momorize feature **Index** 10 and switch to the feature list.



The Feature list.

The feature list, for each feature to which a script/language combination refers, points to one or more lookups via **LookupIndex**. These lookups define the actual substitution and positioning behavior for this feature. Going back to our example, we remember feature index number 10 (the **#** column) and in this feature's **LookupIndex** column click the **+** which opens a nested table pointing to the relevant lookups' indices.



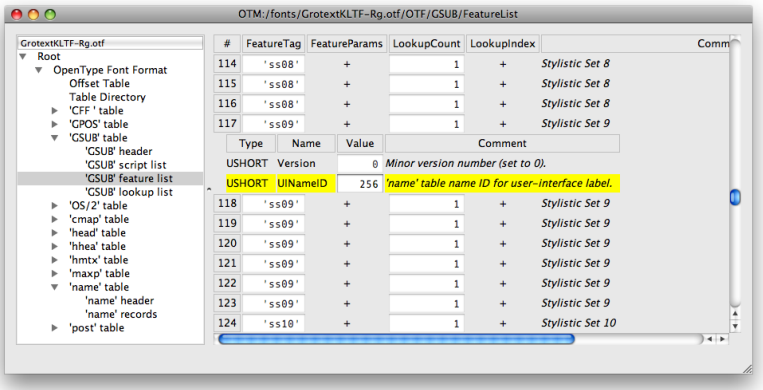
The Lookup list.

The lookup list, finally, holds a list of all lookups. These are presented as nested tables, and you can start unfolding nested tables which strictly represent the lookups' structure. In our example above, there is but a single lookup for kerning, of **LookupType** 2. We click the **+** in the **SubTable** column and get to know about the single subtable's format which is **PosFormat** 1. Clicking the **+** in the **Positioning** column unfolds the single subtable's header information. Clicking the **+** in the **GIDPairValues** row finally presents the kerning pairs and values.

***Note:** The structure of nested tables depends on lookup types and subtable formats – this example really is just that: an example.*

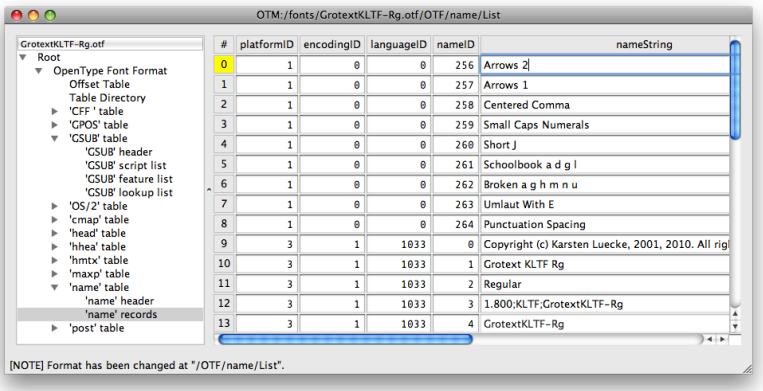
With OTMaster you may view and edit user interface names associated with Stylistic Set 1–20 ('ss01'–'ss20') and Character Variant 1–99 ('cv01'–'cv99') features.

To examine the name of the 'ss09' feature of a given font, choose the 'GSUB' feature list from the table overview. Scroll down to 'ss09' in the content area – there are multiple entries for 'ss09', one per each script/language combination. If there is a **+** in the **FeatureParams** column, then this feature is associated with an interface name. Click the **+** to reveal **Version** and **uiNameID**. The latter is nothing else but a pointer to a nameID associated with one or more name table records.



Locating **uiNameID** *in the* 'GSUB' feature list.

So keep this nameID in mind and choose 'name' records from the table overview. The content area shows, for nameID 256, that this Stylistic Set feature is called 'Arrows 2'. Please note that there may be multiple name records with nameID 256. As usual, there may be different strings for different platforms, encodings, languages.

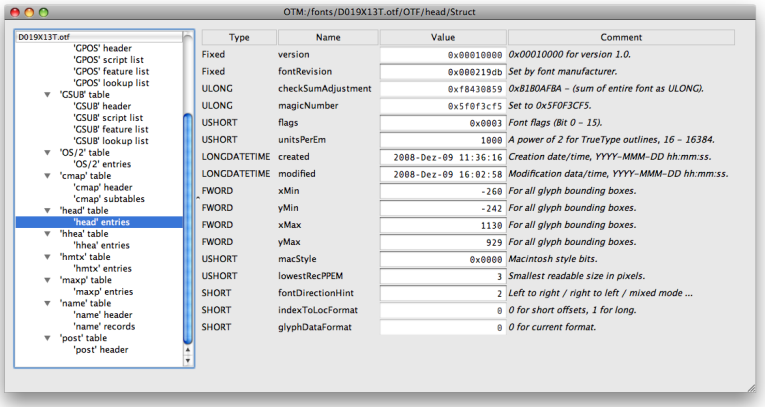


The string associated with this feature found in the name *table.*

— head

The Font Header table holds the most important value – the font's **unitsPerEm** or UPM, i.e. the number of units into which the em-square is divided. All of the fonts metrics relate to it.

Info: The head *table.* ▶*local* ▶*www*



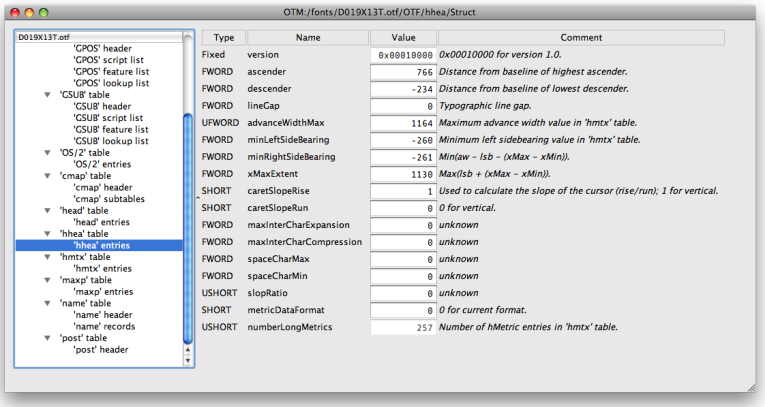
head *table entries.*

macStyle relates to the OS/2 table's **fsSelection** (and **usWeightClass** and **usWidthClass**) and name table records with nameID 2.

— hhea

Many Horizontal Header table entries are not meant to be edited manually.

Info: The hhea *table.* ▶*local* ▶*www*



hhea *table entries.*

caretSlopeRise and **caretSlopeRun** should be in tune with the post table's **italicAngle** value and, if this is a CFF-based OpenType font, also with the 'CFF' top dictionary's **ItalicAngle**.

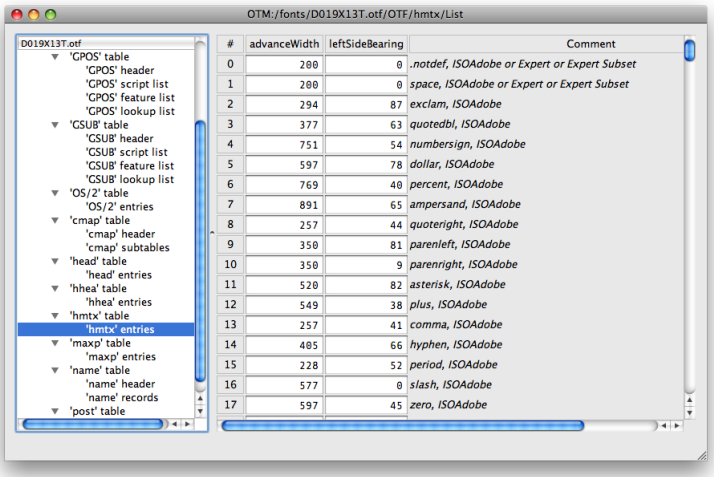
Note: For both italic/slant information and vertical font metrics see the ▶Consistency Checker *chapter.*

ascender, **descender**, **lineGap** relate to OS/2 table's **sTypoAscender**, **sTypoDescender**, **sTypoLineGap**, **usWinAscent**, **usWinDescent**.

— hmtx

The Horizontal Metrics table contains, per glyph, **advanceWidth** and **leftSideBearing**.

Info: The hmtx *table.* ►*local* ►*www*



The basic glyph metrics, advance width and left sidebearing, listed in the hmtx *table.*

In CFF-based OpenType fonts, hmtx table entries are read-only and cannot be edited.

— kern

Info: The kern *table.* ►*local* ►*www*

Theoretically speaking, the Kerning table is a remainder of the (pre-OpenType) TrueType format. Practically speaking, it is still required because many applications cannot read OpenType font's layout tables.

With OpenType fonts, kerning information better be provided by way of a kern feature in the GPOS table. This can deal with 'normal' left-to-right as well as right-to-left and vertical kerning. The specification even says –albeit strangely worded– that CFF-based OpenType fonts are not supposed to have a kern table at all.

Since some applications, even most popular ones, do not support typographic layout features (GSUB, GPOS etc.), unfortunately there is no way around adding a kern table to provide them with kerning information.

However, adding a kern table brings some inconveniences with it. Fonts are getting bigger and in turn require more kerning pairs. For a kern table to be recognized by Windows, it must contain a single subtable of format 0. Since the format 0 subtable's value which indicates the number of kerning pairs can be 65 535 at maximum (not to mention that the value which indicates the subtable's length is similarly restricted),* the number of kerning pairs that may go into such a subtable is limited – not enough to cover all pairs which would result from expanding class-based kerning as found in an OpenType font's GPOS 'kern' feature.

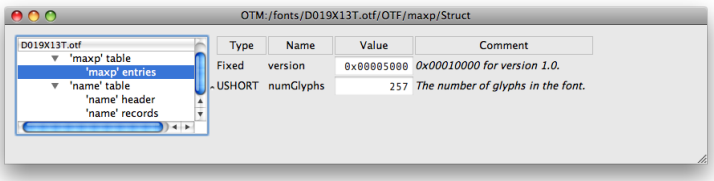
** A detailed description was posted to the OpenType List in 2008 by Ascender Corp.'s Joshua Hadley.*

More about kern table editing in the ►*'kern' Table Viewer* chapter.

— maxp

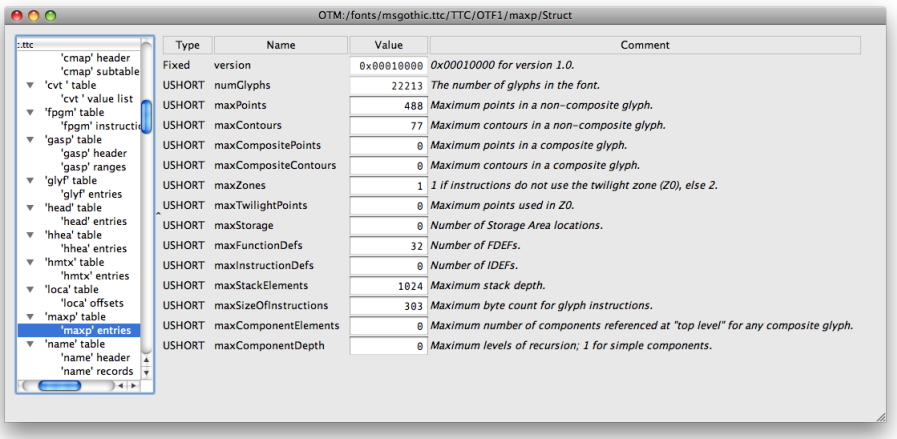
Info: The maxp *table.* ▶ *local* ▶ *www*

With CFF-based OpenType fonts (cubic Bezier curve description in a CFF table), a Maximum Profile table of version 0.5 merely tells about the number of glyphs (**numGlyphs**) in a font.



The version 0.5 maxp *table and its two entries.*

With TT-based OpenType fonts (quadratic Bezier curve description in a glyf table), a Maximum Profile table of version 1.0 holds additional information about glyf, cvt, prep and fpgm tables.



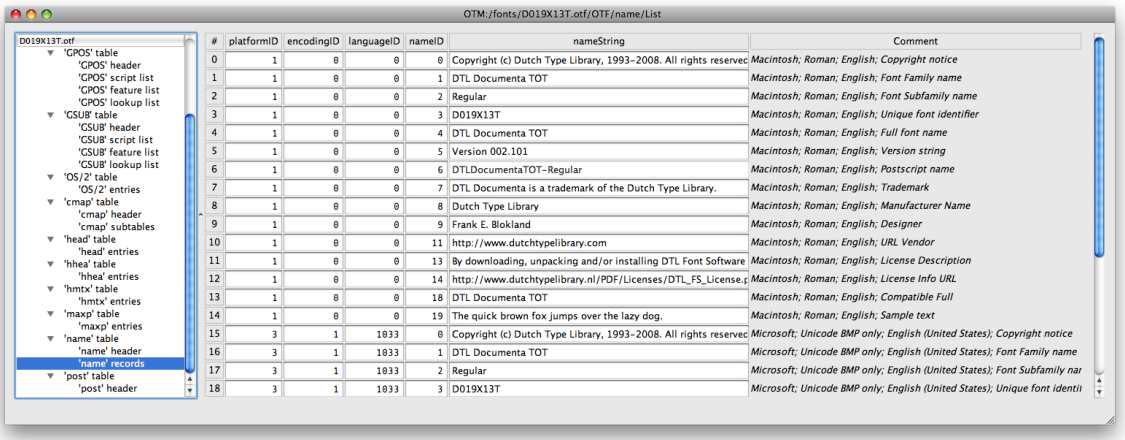
The version 1.0 maxp *table and its additional entries.*

More details about these information and to which tables they relate can be found in Apple's TrueType specification. ▶ *www*

— name

The Naming table is essential for a font to be identified.

Info: The name *table.* ▶*local* ▶*www*



#	platformID	encodingID	languageID	nameID	nameString	Comment
0	1	0	0	0	Copyright (c) Dutch Type Library, 1993–2008. All rights reserved	Macintosh; Roman; English; Copyright notice
1	1	0	0	1	DTL Documenta TOT	Macintosh; Roman; English; Font Family name
2	1	0	0	2	Regular	Macintosh; Roman; English; Font Subfamily name
3	1	0	0	3	D019X13T	Macintosh; Roman; English; Unique font identifier
4	1	0	0	4	DTL Documenta TOT	Macintosh; Roman; English; Full font name
5	1	0	0	5	Version 002.101	Macintosh; Roman; English; Version string
6	1	0	0	6	DTLDocumentaTOT-Regular	Macintosh; Roman; English; Postscript name
7	1	0	0	7	DTL Documenta is a trademark of the Dutch Type Library.	Macintosh; Roman; English; Trademark
8	1	0	0	8	Dutch Type Library	Macintosh; Roman; English; Manufacturer Name
9	1	0	0	9	Frank E. Blokland	Macintosh; Roman; English; Designer
10	1	0	0	11	http://www.dutchtypelibrary.com	Macintosh; Roman; English; URL Vendor
11	1	0	0	13	By downloading, unpacking and/or installing DTL Font Software	Macintosh; Roman; English; License Description
12	1	0	0	14	http://www.dutchtypelibrary.nl/PDF/Licenses/DTL_FS_License.p	Macintosh; Roman; English; License Info URL
13	1	0	0	18	DTL Documenta TOT	Macintosh; Roman; English; Compatible Full
14	1	0	0	19	The quick brown fox jumps over the lazy dog.	Macintosh; Roman; English; Sample text
15	3	1	1033	0	Copyright (c) Dutch Type Library, 1993–2008. All rights reserved	Microsoft; Unicode BMP only; English (United States); Copyright notice
16	3	1	1033	1	DTL Documenta TOT	Microsoft; Unicode BMP only; English (United States); Font Family name
17	3	1	1033	2	Regular	Microsoft; Unicode BMP only; English (United States); Font Subfamily nar
18	3	1	1033	3	D019X13T	Microsoft; Unicode BMP only; English (United States); Unique font identi

The name *table entries. Thanks to the* **Comment** *column, the numeric* platformID, encodingID, languageID *and* nameID *are self-explaining.*

You may remove, edit or add new name entries. To add a name entry, select an existing one and click **Edit > Grow** (⌘+G) (CTRL+G) to duplicate it. Adjust the platformID, encodingID, languageID, nameID and the name tag. Once you have finished adding name entries, click **Edit > Sort** (⌘+R) (CTRL+R) which will reorder all name entries.

Pre-defined nameIDs are, according to the name table specification:

0. *Copyright* notice.
1. *Family* name. Each family, i.e. a collection of fonts which share the same *Family* name, is supposed to consist of no more than these four styles: 'Regular', 'Italic', 'Bold', 'Bold Italic', defined in the *Subfamily* name (2).
2. *Subfamily* (or *Style*) name. For each family –defined by *Family* name (1)– this may be one of 'Regular', 'Italic', 'Bold', 'Bold Italic'. The OS/2 table's **fsSelection** ▶*local* ▶*www* and the head table's **macStyle** ▶*local* ▶*www* need to reflect the style defined by *Subfamily* name.
3. *Unique Font Identifier*. This could be a combination of *Designer* (9) or *Manufacturer* (8) name, *Postscript* name (6) and year, separated by ';'.
4. *Full* name. A combination of *Preferred Family* name (16) and *Preferred Subfamily* name (17) if present or –according to the specification– *Family* name (1) and *Subfamily* name (2). If the Microsoft platform *Subfamily* name (2) is 'Regular', the *Full* name should match this platform's *Family* name (1).
5. *Version* string. This should begin with 'Version [major].[minor]' whereby [major] and [minor] may be any number smaller than 65 535. A *Version* string could look like 'Version 1.500'. Additional information may be added and should be separated by ';'.

Tip: For details see the name *table specification as well as the* ▶Consistency Checker. *Additional information can be found in the* MakeOTF User Guide *which is part of the* AFDKO *download, especially the chapter* New OS/2 Bits. ▶*local* ▶*www Additional* name *table entries and the version 4* OS/2 *table's* ***fsSelection*** *bits 7–9 relate to* WPF*'s font selection model. See Microsoft's paper* WPF Font Selection Model ▶*www for detailed information about* WWS*-conformant families.*

Note: The PostScript *name (6) description mentions an additional condition which the* Full *name (4) needs to meet in* CFF*-based OpenType fonts.*

6. *PostScript* name. This is required for a PostScript interpreter to identify a PostScript language font corresponding to this OpenType font. There must be a record with nameID 6 for both (platformID–encodingID–languageID):

1–0–0 (Macintosh)
3–1–1033 (Microsoft)

Both strings must be identical when translated to ASCII and not longer than 63 characters. Characters are restricted to ASCII codes 33–126 excluding any of '**[](){}<>/%**'. Usually built like the *Full* name (4) – but without any spaces and with with a '**-**' between family and subfamily part of the string.

There is an additional requirement for CFF-based OpenType fonts: The *PostScript* name (6) for both Microsoft platform and Macintosh platform, the *Full* name (4) for the Microsoft platform as well as the CFF Name INDEX must be identical. OTMaster displays the CFF Name INDEX as **FontName** inside of the 'CFF' top dictionary, thereby deviating a bit from the CFF table's structure.

7. *Trademark* notice.
8. *Manufacturer* name.
9. *Designer* name.
10. *Description* of the typeface.
11. *Vendor* URL. This should include 'http://', 'ftp://' etc.
12. *Designer* URL. This should include 'http://', 'ftp://' etc.
13. *License* description. This is expected to be a summary of the terms rather than, as the specification puts it, 'legalese'.
14. *License Info* URL. A link to the full license text. This should include 'http://', 'ftp://' etc.
15. Reserved, set to zero.
16. *Preferred Family* name. If a family consists of more, or other, styles than 'Regular', 'Italic', 'Bold', 'Bold Italic', you may define a *Preferred Family* name for which there are no restrictions as to the number of styles.
17. *Preferred Subfamily* (or *Style*) name. This relates to the *Preferred Family* name (16). If the OS/2 table's version is 4 and if a family's *Preferred* styles are wws-conformant (i.e. are distinguished by the categories weight, width and slope alone), you may set the OS/2 table's **fsSelection** bit 8 to 1 which will signal wws-conformancy. If a family's *Preferred* styles are not wws-conformant, set bit 8 to 0 and provide wws-conformant *wws Family* name (21) and *wws Subfamily* name (22).

*Note: If you set the OS/2 table's version to 4, then all the **fsSelection** bits 7–9 need to be set to 0 or 1 consciously because starting with this table version, these bit settings bear a special meaning.*

18. *Compatible Full* name. Macintosh platform only. If the *Full* name (4) turns out to be too long, i.e. longer than 27 (or more generously: 31) characters, you may either abbreviate the Macintosh platform *Full* name (4) or otherwise provide an additional abbreviated *Compatible Full* name. If the *Compatible Full* name is present and the *Subfamily* name (2) is 'Regular', the *Compatible Full* name should match the *Family* name (1).
19. *Sample* text.
20. *PostScript* CID findfont name. Please see the name table specification.

21. *wws Family* name. A wws-conformant family's styles must address no other categories than weight, width and slope. If this name is provided, the OS/2 table's **fsSelection** bit 8 should be set to 0.
22. *wws Subfamily* (or *Style*) name. If this name is provided, the OS/2 table's **fsSelection** bit 8 should be set to 0. The OS/2 table's **usWeightClass**, **usWidthClass** as well as **fsSelection** bits 0 (italic) and 9 (oblique) should reflect the *wws Subfamily* name.

The name table specification ▶*local* ▶*www* comes with an example string for each nameID, so that you are encouraged to read this documents carefully.

Some OpenType layout features like Stylistic Set and Character Variant features can be associated with strings that applications may display in their user interfaces. These strings are located in the name table and referenced from the GSUB table. Please see the ▶*GPOS/GSUB* section for an example.

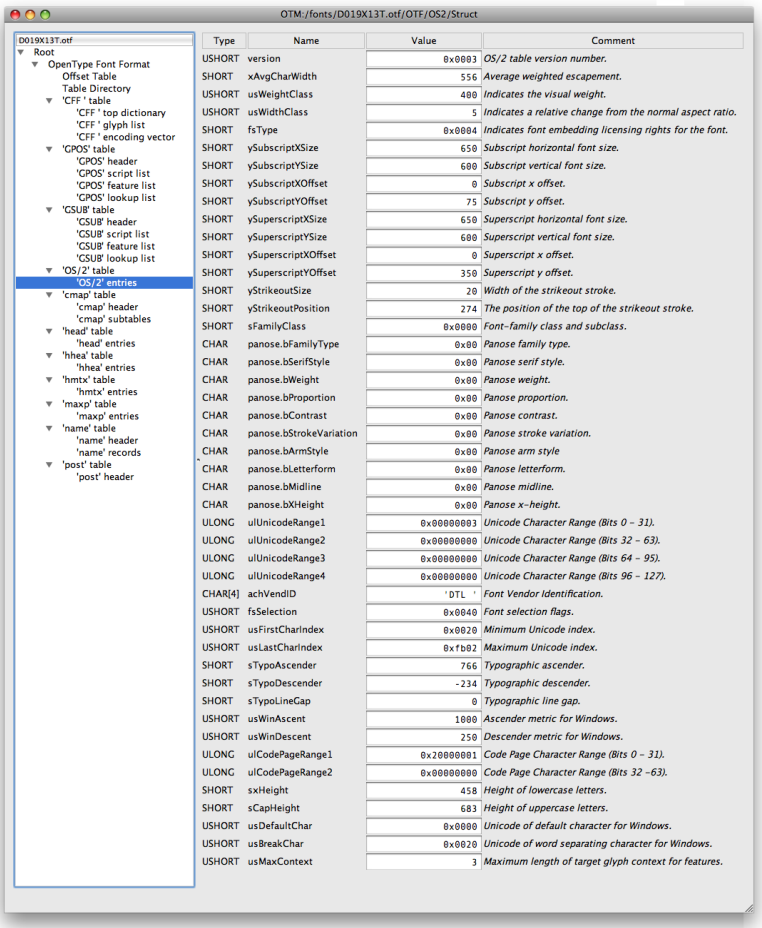
Note: A wws-conformant font may have both Italic and Oblique styles! See Microsoft's paper WPF Font Selection Model. ▶*www*

Note: If you set the OS/2 *table's version to 4, then all the* ***fsSelection*** *bits 7–9 need to be set to 0 or 1 consciously because starting with this table-version, these bits bear a special meaning.*

— OS/2

Most font-wide information are located in the OS/2 table. The content area's **Comment** column provides a description for every entry.

Info: The OS/2 *table.* ▶*local* ▶*www*



usWeightClass, **usWidthClass** and **fsSelection** relate to the name table records with nameIDs 1/2, 16/17, 21/22, while **fsSelection** also relates to the head table's **macStyle**. Please see the ▶*name* table section.

fsType defines embedding licensing rights for this font. May it be embedded into a document? What, then, is allowed with such a document?

sTypoAscender, **sTypoDescender**, **sTypoLineGap**, **usWinAscent**, **usWinDescent** –and the hhea table's **ascender**, **descender**, **lineGap**– are dealt with in the ▶*Consistency Checker* chapter.

If you set the OS/2 table's version to 4, you need to set **fsSelection** bits 7–9 to 0 or 1 consciously because starting with this table version, each of these bits carries a special meaning:

7. To tell applications that the default line-to-line distance should be calculated from **sTypoAscender/sTypoDescender/sTypoLineGap**, set this bit to 1. Otherwise, set this bit to 0. See the ▶ *Consistency Checker* chapter for more information about setting vertical font metrics.

8. If this family is wws-conformant (the name table *Preferred Subfamily* names (17) address no other categories than weight, width and slope), set this bit to 1. If this is not so (e.g. if *Preferred Subfamily* names (17) refer to optical size or even something like 'Outline'), set this bit to 0 but add wws-conformant *wws Family* name (21) and *wws Subfamily* name (22).

9. If this font's slope can be described as 'Oblique' rather than 'Italic' –WPF distinguishes between 'Oblique' and 'Italic'–, set this bit to 1. Otherwise, set this bit to 0. (If the font is 'Italic', set bit 0 to 1 as usual.)

Tip: See Microsoft's paper WPF Font Selection Model, *especially pp. 4–6 and the 'Guidelines for font manufacturers' on pp. 17–18.* ▶ *www*

If you set the OS/2 table's version to 5, you will notice two additional entries. These allow to create optical size specific fonts and define the intended size range for which a font was designed. Please note that these entries interact with name table records. Microsoft, who came up with this, has not published a specification about the details yet.

USHORT	version	0x0005	*OS/2 table version number.*
USHORT	usLowerPointSize	0	*Lower value of size range for multiple optical styles.*
USHORT	usUpperPointSize	65535	*Upper value of size range for multiple optical styles.*

usLowerPointSize is the lower value of the size range for which this font is to be used while **usUpperPointSize** is the upper value of the size range at which the font is to be used. One font's **usLowerPointSize** of one optical size may be identical to another font's **usUpperPointSize** since one of both values will be excluded from the range.

One unit corresponds to 0.05 pt, or 1 twip.

For font families that do not feature special designs for separate point size ranges, **usLowerPointSize** should be 0 and **usUpperPointSize** should be 65535 which is an infinitely large size.

Info: Preliminary information on these new values by John Hudson. ▶ *www*

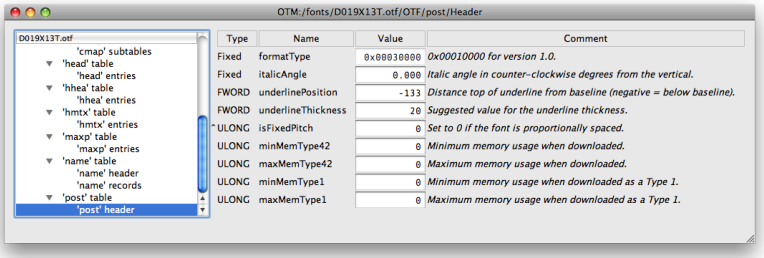
Note: ***usLowerPointSize*** *nad* ***usUpperPointSize*** *unit is a 'twip' which equals 0.05 pt.*

— post

Info: The post *table.* ▶ *local* ▶ *www*

The post table holds information for printing fonts on PostScript printers. TT-based OpenType fonts store glyph names in a post table of version 2.0. CFF-based OpenType fonts, unless they are CID-keyed, store glyph names in the CFF table. In this case, the post table has format 3.0 which signifies an abbreviated version of this table that omits glyph name information. It is possible for TT-based OpenType fonts, too, to make use of a post table of version 3.0. In this case, the font does not contain glyph names at all.

Changing the **formatType**, i.e. the table's version number, from 2 to 3 will remove glyph names from the table. Changing it from 3 to 2 will add glyph names (placeholders, actually).



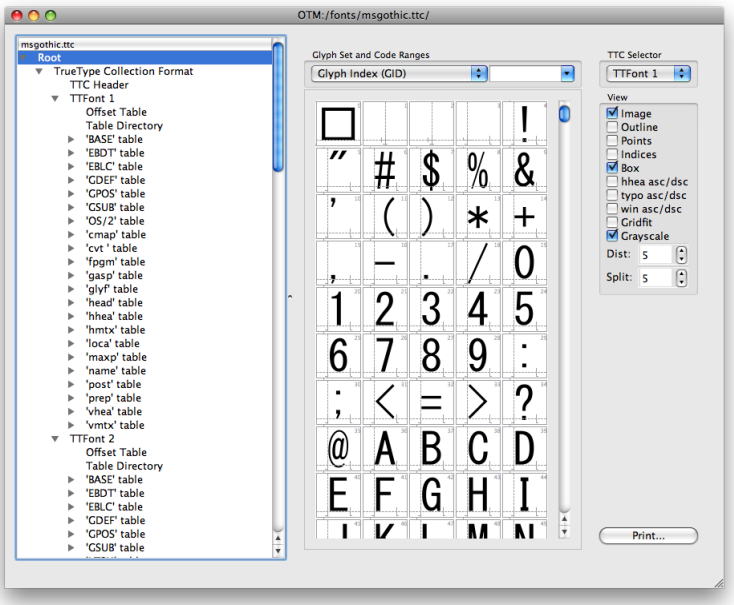
A version 3.0 post *table.*

The **italicAngle** value should be consistent with the **ItalicAngle** in 'CFF' top dictionary (if this is a CFF-based OpenType font) and with the hhea table's **caretSlopeRise** and **caretSlopeRun**. More in the ▶ *Consistency Checker* chapter.

If the font is a CFF-based OpenType font, make sure **underlinePosition** and **underlineThickness** are consistent with the 'CFF' top dictionary's **UnderlinePosition** and **UnderlineThickness**. The underline position is calculated differently in both tables, please see the ▶ *CFF* section for details.

TTC Fonts

A TTC (TrueType Collection) font contains more than one sub-font. A TTC font's TTC Header points to each sub-font's header information consisting of Offset Table and Table Directory which points to tables associated with this sub-font. (Version 2.0 of the TTC Header will also reference a common DSIG table.) These then are followed by all tables. This allows a TTC's sub-fonts to 'share' tables that are identical, and – which is the example given in the OpenType specification – even use a single glyf table holding all glyphs of all sub-fonts, yet each sub-font's loca table only refers to glyphs relevant for this sub-font.

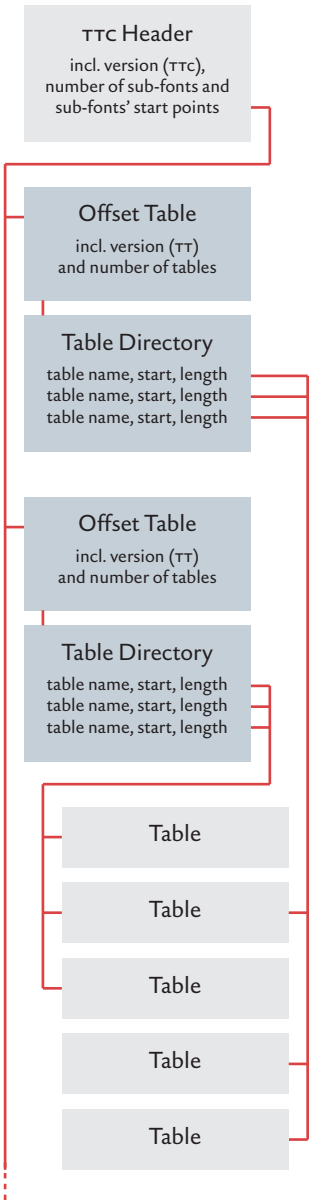


With TTC fonts, OTMaster's table overview starts with the Root entry. Root holds TrueType Collection Format which in turn holds TTFont 1, TTFont 2, TTFont 3, etc., as many as there are sub-fonts in the TTC font. Each TTFont (like the OpenType Font Format) holds an Offset Table and Table Directory, followed by all tables found in the sub-font.

OTMaster lists, for each sub-font, all referenced tables – even if these are 'shared' by sub-font. Once that a 'shared' table is adjusted for a single font, this table will be duplicated.

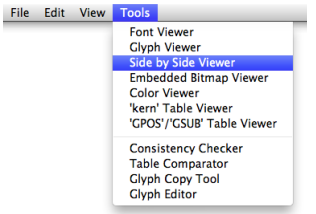
Also see *The OpenType Font File.* ►*local* ►*www*

The structure of a TTC font's data (simplified):



OTMaster's Toolbox

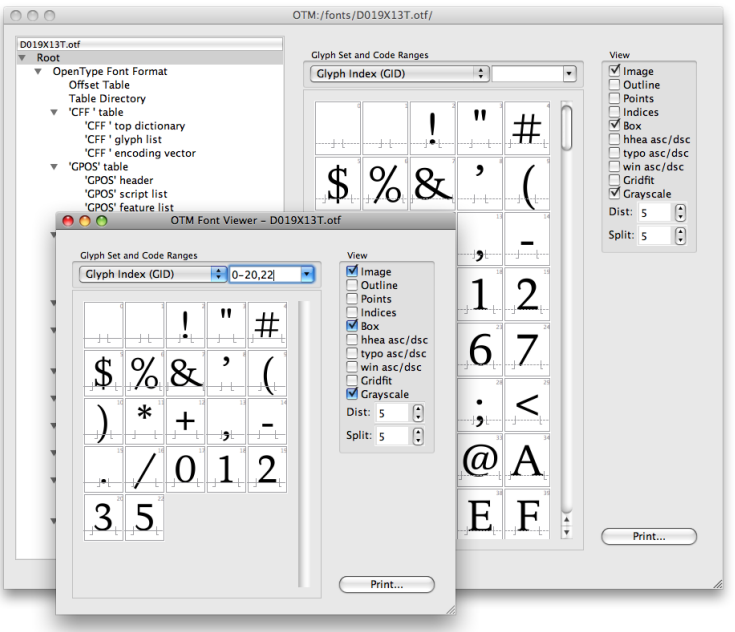
OTMaster is not only good for reviewing and editing a font's 'raw' tables but comes with tools for higher-level – and thus more user-friendly – access to parts of a font's data. Some of these tools do not show a specific table's content but represent data in a thematic/categorized way: The Font Viewer shows all, or a selection, of a font's glyphs. The Glyph Viewer shows all data related to an individual glyph, thereby combining data from various tables like glyf or CFF for outlines, post or CFF for glyph names, cmap for Unicode codepoints, hmtx and/or vmtx for metrics, etc.



TOOLS MENU

Font Viewer

The table overview's Root and the Font Viewer alike give a visual summary of a font's glyph set:



When selecting Root, the content area shows the same glyph overview as does the Font Viewer.

— Glyph Set and Code Ranges

These define which part of the glyph set to show, and how:

The leftside popup menu offers categories for defining glyph sub-sets by **Glyph Index (GID)**, **Unicode**, or any cmap subtable present in the font. The category also determines how glyphs are sorted in the overview.

The rightside textbox allows you to restrict the glyph overview to selected glyphs or glyph ranges. These are defined by GID or Unicode code-point, depending on the category selected in the popup to the left.

*Glyph indices are expected to be integer numbers (0, 1, 123). Unicode codepoints are expected to be hexadecimal numbers, without trailing zeros but preceded by '**0x**' (0x31, 0xB5).*

You can provide a single identifier (0x20), a comma-separated list of identifiers (0x20, 0x61), a range delimited by first and last identifier (0x20-0x61), or a combination thereof (0x20, 0x40-0x61, 0xB5).

You need to confirm with ⏎.

The text input box will remember previous glyph sub-sets, and you may get back to them by using the boxes' popup functionality – by clicking on the popup menu's arrow button or by placing the cursor inside the box and using ↑ or ↓ to jump to any previous glyph sub-set.

— View

Image shows filled shapes.
Outline adds a colored outline.
Points shows on-curve points.
Indices shows points' index numbers.
Box draws the bounding box around each glyph.
hhea asc/dsc are the hhea table's ascender/descender heights.
typo asc/dsc are the OS/2 table's sTypoAscender/sTypoDescender height.
win asc/dsc are the OS/2 table's usWinAscent/usWinDescent heights.
Gritfit will align points, at current ppem size, to the pixel grid.
Grayscale draws grayscaled outlines.
Dist. is the distance between bounding boxes. With a value of 0, bounding boxes will touch.
Split determines the number of glyphs per line.

*Note: Since OTMaster uses the FreeType rasterizer, **Grayscale** may produce results that differ from 'real world' output, especially at small ppem sizes.*

A glyph index is shown in the top left corner of each glyph's bounding box.

A double-click on a glyph in the Font Viewer glyph overview, or in the Root content area, will open this glyph in the ▶ *Glyph Viewer*.

⇧ + double-click on a glyph will open it in the ▶ *Glyph Editor*.

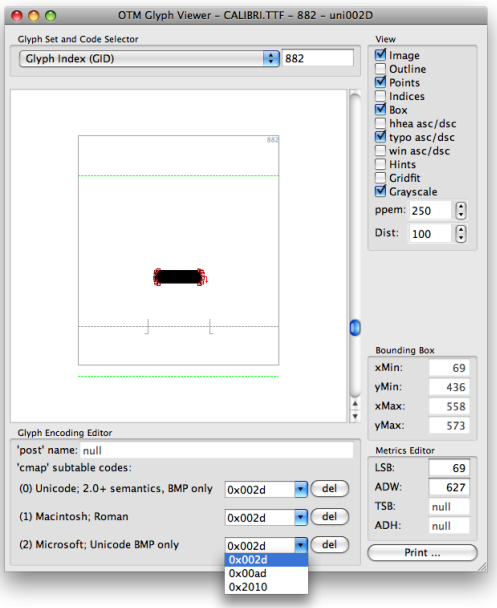
Glyph Viewer

Per glyph, the Glyph Viewer interface gives access to all glyph-related data.

— Glyph Set and Code Selector

In the left popup menu choose whether to identify a glyph by **Glyph Index (GID)** or **Unicode** codepoint. In the right text input box, define a glyph's glyph index or codepoint.

*Glyph indices are expected to be integer numbers (0, 1, 123). Unicode codepoints are expected to be hexadecimal numbers, without trailing zeros but preceded by '**0x**' (0x31, 0xB5).*



Use the scrollbar to the right of the glyph review area to flip through all glyphs!

— View

Image shows filled shapes.
Outline adds a colored outline.
Points shows on-curve points.
Indices shows points' index numbers.
Box draws the bounding box around each glyph.
hhea asc/dsc are the hhea table's ascender/descender heights.
typo asc/dsc are the OS/2 table's sTypoAscender/sTypoDescender height.
win asc/dsc are the OS/2 table's usWinAscent/usWinDescent heights.
Hints shows hints if existing.
Gritfit will align points, at current ppem size, to the pixel grid.
Grayscale draws grayscaled outlines.
ppem is the ppem size at which the glyph is rasterized.
Zoom allows you to scale the rasterized image up or down which makes it possible to see the actual rasterization in detail.

***Note:** Since OTMaster uses the FreeType rasterizer, **Grayscale** may produce results that differ from 'real world' output, especially at small ppem sizes.*

***Tip:** If a font's UPM (units per em) is 1000 and if you choose 160 **ppem**, then the 1000 units will be scaled such that they fit into 160 pixels upon glyph rasterization. The effect of this can be studied by increasing the **Zoom** factor.*

— Glyph Encoding Editor

The first entry is the glyph name, which is called
1. **'CFF' id** with CFF-based OpenType fonts ('id' because in CID-keyed fonts this would not be a name but a mere index value), and
2. **'post' name** with TT-based OpenType fonts.

'cmap' subtable codes shows as many entries as there are subtables in the cmap table. For each subtable there is a popup text box which may contain none, one or more Unicode codepoints. Click the **del** button to delete the respective subtable's codepoints.

Changes in this area will be reflected in 'cmap' table and vice versa.

Tip: Make sure that you provide or adjust codepoints for all cmap subtables, not just one …

— Bounding Box

xMin is the smallest horizontal extension of the glyph.
yMin is the smallest vertical extension of the glyph.
xMax is the largest horizontal extension of the glyph.
yMax is the largest vertical extension of the glyph.

Note: In CFF-based OpenType fonts, these values relate to smallest or largest extensions of a glyph regardless if there are on-curve extremum points or not. In TT-based OpenType fonts, these values relate to coordinates of outermost on-curve or off-curve points.

— Metrics Editor

LSB is the left sidebearing which is equal to xMin.
ADW is the advance width.
TSB is the top bearing.
ADH is the advance height.

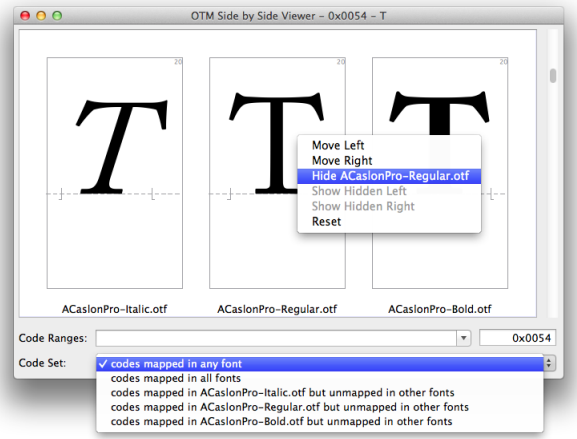
The right sidebearing is implicit and can be calculated as

RSB = ADW – xMax

TSB and **ADH** relate to vertically oriented glyphs whose origin point, in CFF-based OpenType fonts, is at the top center of a glyph's bounding box. An illustrations for this can be found in Adobe's AFM specifications, p. 11, fig. 1. ▶ *www*

Side by Side Viewer

The Side by Side Viewer shows, for a given character, the corresponding glyphs from all fonts that are currently opened in OTMaster.



Use the scrollbar to the right of the review area to navigate through the available or selected range of characters.

As with other viewers too, use the scrollbar to navigate through the character set.

Code Ranges allows you to select a range of characters offered for review by stating single, or ranges of, Unicode codepoints.

Code Set helps you narrow down the selection of characters offered for a visual comparison: **codes mapped in any font** shows characters even if they do not exist in one or more of the fonts, **codes mapped in all fonts** shows characters only if present in all fonts, **codes mapped in […] but unmapped in other fonts** shows all characters present in the according font even if they do not exist in onr or more of the other fonts.

*Unicode codepoints are expected to be hexa-decimal numbers, without trailing zeros but preceded by '**0x**' (0x31, 0xB5).*

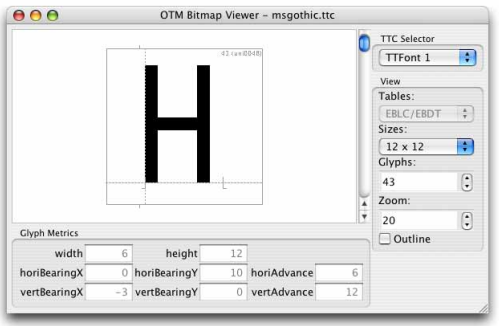
You can provide a single identifier (0x20), a comma-separated list of identifiers (0x20, 0x61), a range delimited by first and last identifier (0x20-0x61), or a combination thereof (0x20, 0x40-0x61, 0xB5).

Confirm with ↵.

A right-click into a specific glyph's box brings up a context menu whose functions help rearranging the displayed glyph. It can be moved to the left or right or can be hidden, and shown again. The manual arrangement can be reset to restore the default arrangement.

Embedded Bitmap Viewer

The Embedded Bitmap Viewer visualizes bitmaps from EBLC and EBDT or other bitmap tables.



The Embedded Bitmap Viewer. Use the scrollbar to the right of the glyph review area to flip through all glyphs!

— View

Tables allows you to select which tables' bitmap information to show.
Sizes are the bitmap sizes covered by the selected tables.
Glyphs allows choosing a glyph by its glyph index.
Zoom will zoom in or out.
Outline displays the outline as well.

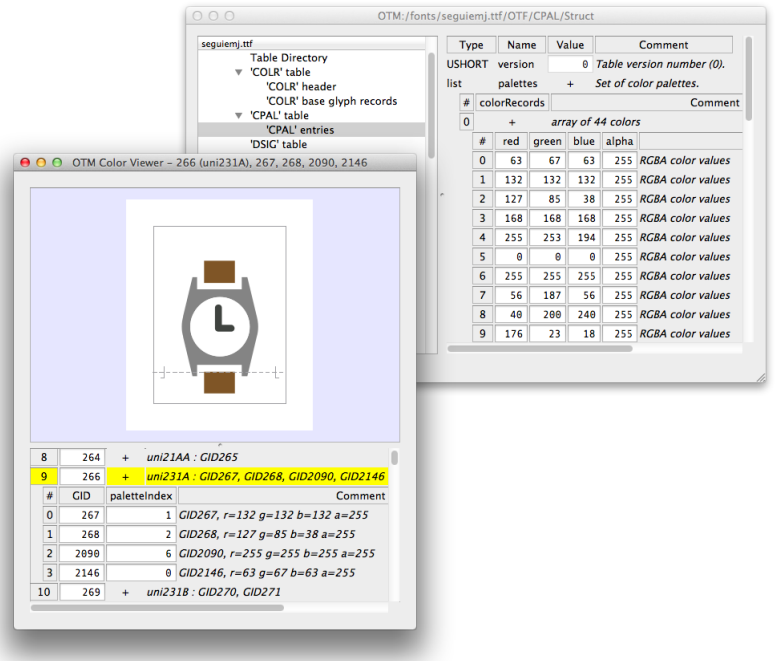
— Glyph Metrics

width is the number of columns of data, i.e. the number of pixels in horizontal direction.
height is the number of rows of data, i.e. the number of pixels in vertical direction.
horiBearingX is the number of pixels from the origin to the left edge of the bitmap (horizontal layout).
horiBearingY is the number of pixels from the origin to the top edge of the bitmap (horizontal layout).
horiAdvance is the bitmap's advance width (horizontal layout).
vertBearingX is the number of pixels from the origin to the left edge of the bitmap (vertical layout).
vertBearingY is the number of pixels from the origin to the top edge of the bitmap (vertical layout).
vertAdvance is the bitmap's advance height (vertical layout).

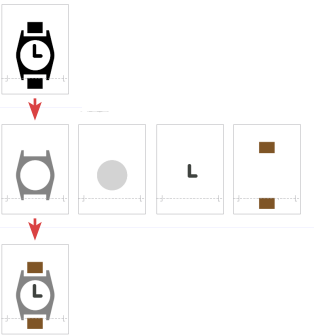
All these values are read-only.

Color Viewer

In summer 2013, Microsoft presented their solution for color fonts and introduced two new tables, COLR and CPAL. The solution is elegantly simple. Starting point is a base glyph which usually is encoded and serves as a fallback for applications that do not know the new tables. The COLR table maps this to-be-colored base glyph to one or more glyphs, each of which holds outlines for a specific color plus references to the respective color definitions. The CPAL table in turn provides the color definitions, stored as RGBA values. There may be multiple sets of color definitions, called 'palettes', from which applications may choose.



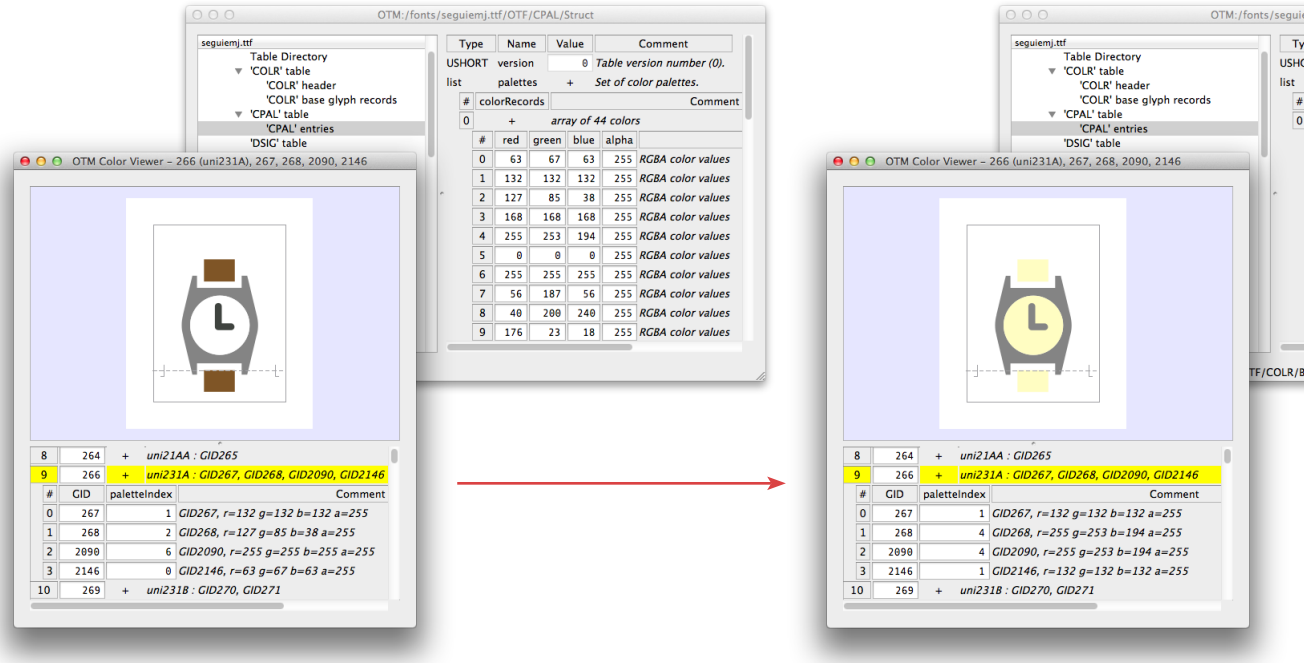
This structure is reflected in the Color Viewer which essentially visualizes the content of the COLR table. The top level is a list of base glyphs, identified by **GID**. Clicking the **+** sign next to one of them will reveal a nested list of pairs each consisting of one color-specific glyph, again by **GID**, and a reference to a color definition, by **paletteIndex**. The latter in turn refers to a **colorRecord** stored in the CPAL table; see the **#** column for the **paletteIndex**. Please note that there may be multiple color **palettes** in the CPAL table, each of which has its own definition for a given **colorRecord** by way of **red**, **green**, **blue** and **alpha** values. A special **paletteIndex** value of 0xFFFF indicates that the user-defined foreground color is to be used.



Top: Base glyph 266. COLR will map it to color-specific glyphs. Middle: Color-specific glyphs. Glyph 267 for grey outlines, glyph 2090 for white outlines, glyph 2146 for dark grey outlines, glyph 268 for brown outlines.
Bottom: Colored glyphs combined.

Open the Color Viewer and ***'CPAL' entries*** *side by side to review both a color glyph's composition and the colors it makes use of.*

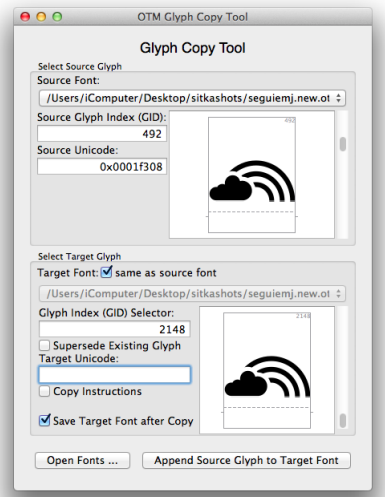
There is more than one way to change a color: Either adjust a **paletteIndex** in the Color Viewer, thereby selecting another predefined color. Or open the CPAL table in OTMaster's main dialog, choose a palette, then redefine the **red**, **green**, **blue** and **alpha** values of a given **colorRecord**. Or add a new **colorRecord** by duplicating an existing one, adjusting its values, and referring to it from the Color Viewer.



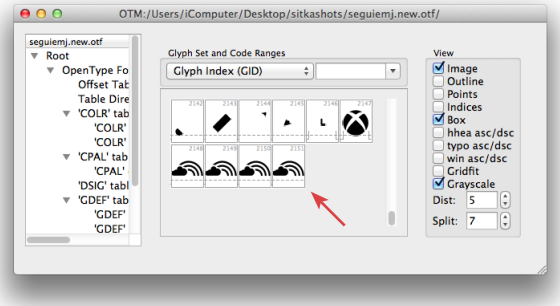
Creating a color glyph from a normal glyph takes only a few steps:

1. Open the Glyph Copy Tool. Choose a base glyph, either by entering a numeric **Source Glyph Index (GID)** or by using the scrollbar next to the glyph image. For every color, make a copy of this glyph by clicking the **Append Source Glyph to Target Font** button. **Target Font** should be **same as source font**. Please note that glyph copies will be added after the (target) **Glyph Index (GID) Selector**.

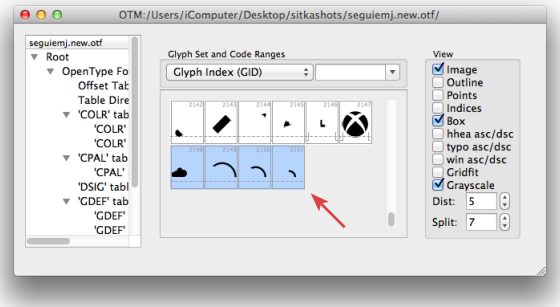
To add new glyphs at the end of the font, enter any number higher than the number of glyphs in the font. OTMaster will automatically make it the last GID in the font.

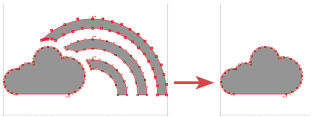


In our example, we appended new glyphs at the end of the font:



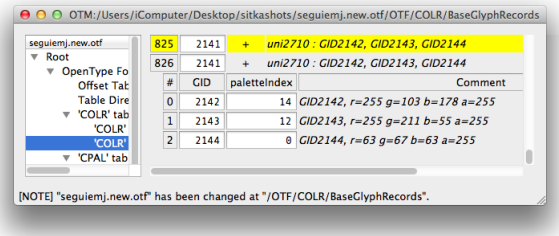
2. Open each of the new glyphs in the Glyph Editor, choose the **Shift** tool, select and delete all contours not needed in this glyph. The result:





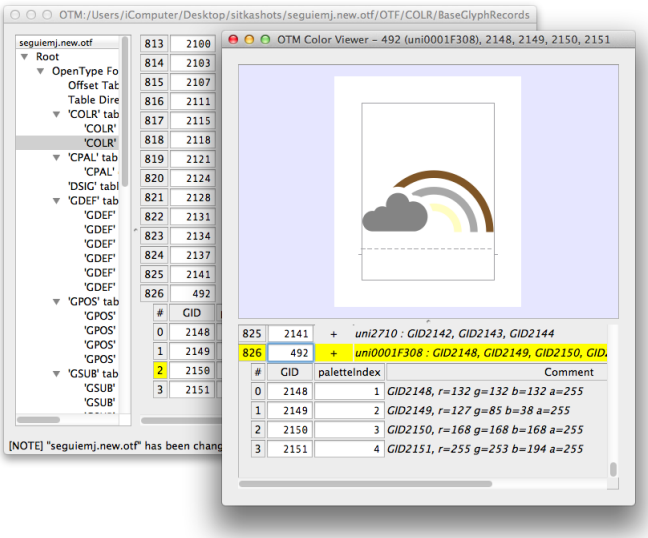
Deleting contours not needed in a specific color glyph via the **Shift** *tool.*

3. Go to the COLR table's content, select any entry and use the Edit menu's **Grow** function. This will duplicate the selected entry which serves as a template for our new color glyph definition. Below, we chose entry number 825 to create the new entry number 826. (If there are no COLR and CPAL tables in our font, you may **Cut** and **Paste** them over from another font.)



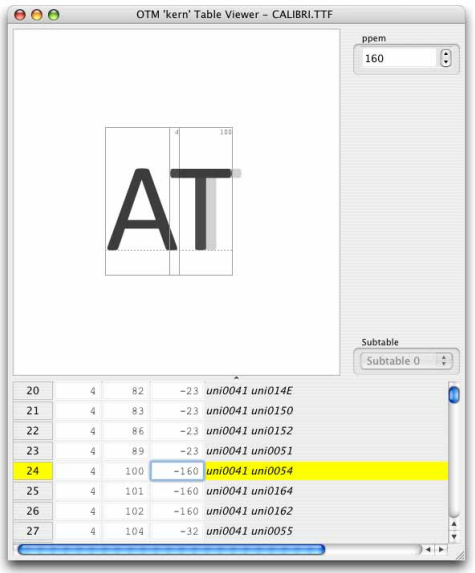
After opening the nest by clicking the **+** sign, you'll see entries for specific color glyphs. If there are too many, select and **Cut** those not needed. If there are not enough, select and **Grow** any of them. In our example, we need four entries – one for the cloud, three for the rainbow elements. Now we adjust GID numbers. The top-level GID will reflect the base glyph's 492, the nested color glyph GIDs will reflect the new glyphs' 2148–2151.

4. Open the Color Viewer. Scroll down to the COLR table entry created and adjusted in the last step. Change colors by adjusting the **paletteIndex** for each specific color glyph's **GID**. Below, we assigned color definitions 1–4.



'kern' Table Viewer

This is a most useful tool to visually check and adjust the kern table. The currently selected pair is presented in main part of the window.



The 'kern' Table Viewer dialog.

— ppem

This single option serves to enlarge or reduce the size of the kerning pair, by choosing another ppem size.

— Subtable

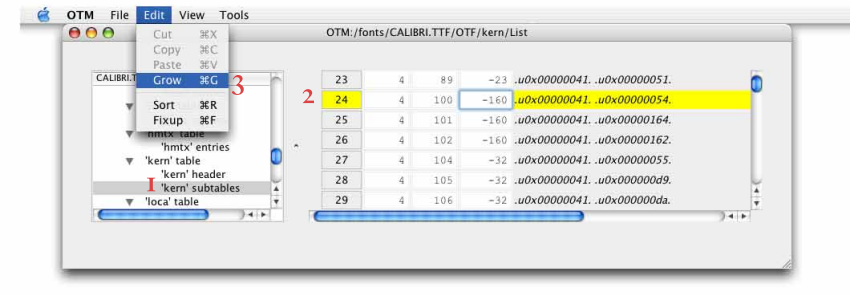
Here you may select which subtable's kerning you want to review or edit.

Note: A kern *table may contain more than one subtable, but please be aware that Windows and* OS/2 *accept a* kern *table only if it holds a single subtable of format 0. Also see the* kern *table specification.* ►*local* ►*www*

The list of kerning pairs which follows consists of three editable columns:
left glyph of the pair by glyph index,
right glyph of the pair by glyph index,
and the kerning **value** which is relative to the font's UPM value.
In addition, a **Comment** column translates left and right glyphs' indices into glyph names as found in either CFF or post table.

Like in **Nested Tables** viewing mode, use ↑ (arrow up) and ↓ (arrow down) keys to flip through all kerning pairs quickly, and ⇥ (to right) and ⇧+⇥ (or ⇤; to left) to jump from column to column.

Existing kerning pairs –**left** and **right** glyphs' indices as well as the kerning **value**– can be reviewed and adjusted in the 'kern' Table Viewer, but it is not possible to remove or add kerning pairs. With a little trick, though, you may do so:



Remove or add a kerning pair in three steps.

1. Go to the main window and select 'kern' subtables (inside of 'kern' table) from the table overview. Click the **+** sign next to the subtable which you intend to edit, this will fold out the subtable as a nested table. Possibly double-click the subtable's header to show its entries in a new window.

Now either remove a kerning pair:

2. Select the pair which you intend to remove.
3. Choose **Edit > Cut** (⌘+X) (CTRL+X).

Or add a new kerning pair:

2. Select any kerning pair, preferrably the last one.
3. Choose **Edit > Grow** (⌘+G) (CTRL+G) to duplicate this selected pair.

Change **left** glyph index, **right** glyph index and/or kerning **value**, and you have added a new kerning pair! Go back to the 'kern' Table Viewer and visually adjust the kerning **value**. If you added new pair(s) at the end of the kern table, you know where to find them …

Finally choose **Edit > Fixup** (⌘+F) (CTRL+F) to reorder kerning pairs by glyph indices.

'GPOS'/'GSUB' Viewer

The 'GPOS'/'GSUB' Viewer is a tool for reviewing the content of the two most important OpenType layout tables, GSUB for glyph substitution and GPOS for glyph positioning. In case of the GPOS table, positioning values can be adjust. The viewer presents these tables' data visually, which makes it possible to check a font's layout behavior in a comfortable way. Data is, structurally, presented in the same way as found in these tables.

Both GSUB and GPOS tables consist of three lists: Script list, Feature list, and Lookup list.

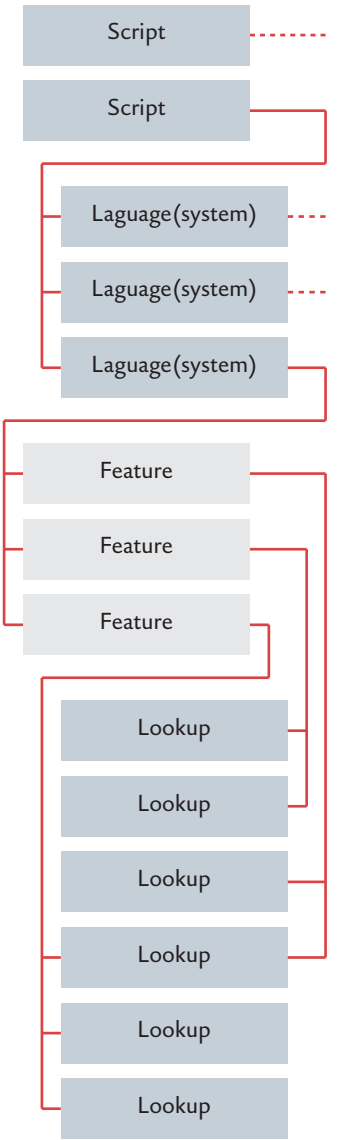
The Script list sums up all scripts explicitly addressed by the layout table (these are scripts – or writing systems – like 'latn' for Latin or 'cyrl' for Cyrillic).* Per each script, there is a list of languages explicitly addressed by the layout table as well as a 'default' language for which there is a special place in the data structure. In case that a layout application cannot find a match for the selected language (e.g. by way of the spelling dictionary in Adobe InDesign) in the font's layout table, it would fall back to this 'default' language. And per each language there is a list of features associated with it.

The Feature list, for each feature to which a script/language combination refers, points to one or more lookups in which the actual substitution and positioning behavior is defined.

Finally, the Lookup list points to individual lookups, each of which defines portions of layout behavior.

Users of VOLT are familiar with this structure because VOLT presents layout data in a way which resembles layout tables' data structure. Users of AFDKO and Adobe's feature file syntax may need some time to get accustomed to it because the higher-level nature of Adobe's feature file syntax hides the complexity of the data structure of compiled layout tables. So again our recommendation that you study especially the documents related to OpenType layout tables, in particular to GSUB and GPOS.

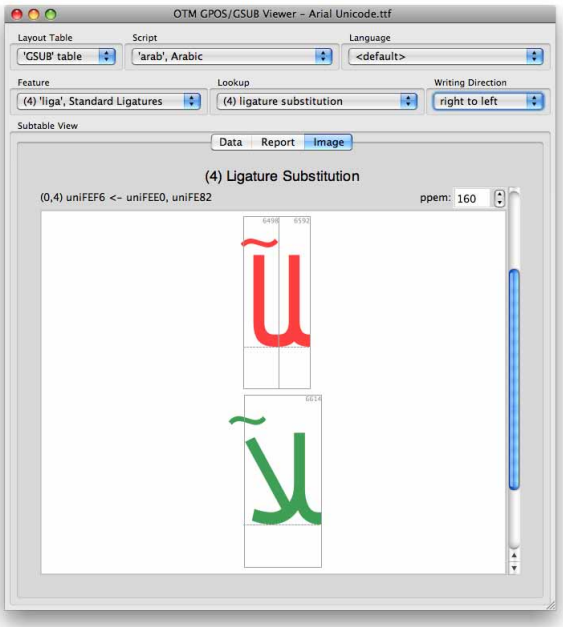
OpenType layout data is organized by script, language system, typographic feature and lookup:

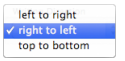


* *The default script, or 'DFLT' (all-caps!), was introduced rather late by Adobe. While there is a special place for the default language, or 'dflt' (lowercase!), in the Language list, there is no such thing for the 'DFLT' script – hence it is a script like any other, included in the regular Script list, and identified by its tag.*

Info: *The* GPOS *table.* ►*local* ►*www* *The* GSUB *table.* ►*local* ►*www* *Also see the* OpenType Layout Common Table Formats document. ►*local* ►*www*

The 'GPOS'/'GSUB' Viewer's top popup boxes reflect the layout tables' internal structure:





***Writing Direction** options.*

— Layout Table

At first you need to choose from 'GSUB' table or 'GPOS' table – 'GSUB' table cares for glyph substitution, 'GPOS' table cares for glyph positioning.

— Script

Select the script whose lookups you plan to review.

— Language

Select the language whose lookups you plan to review. The choise of languages depends on which **Script** you have selected previously.

— Feature

Select a feature whose lookups you plan to review. The choice features depends on which **Language** you have selected previously.

— Lookup

The selection of lookups shown in this popup box is determined by your previous choices of **Script**, **Language** and **Feature**.

*Tip: To see more lookups, and independently of previous choices of **Script**, **Language** and/or **Feature**, select the option <any> in **Script**, **Language** and/or **Feature**.*

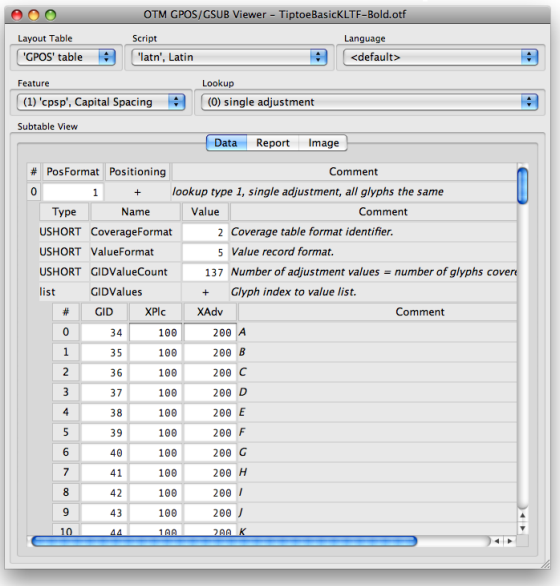
— Writing Direction

Options **left to right**, **right to left** and **top to bottom** make sure that glyph strings are presented in correct order.

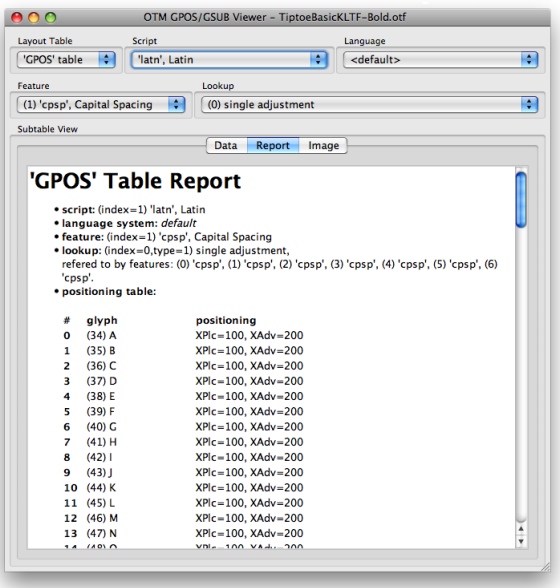
— Subtable View

The content of the lookup can be viewed in three modes.

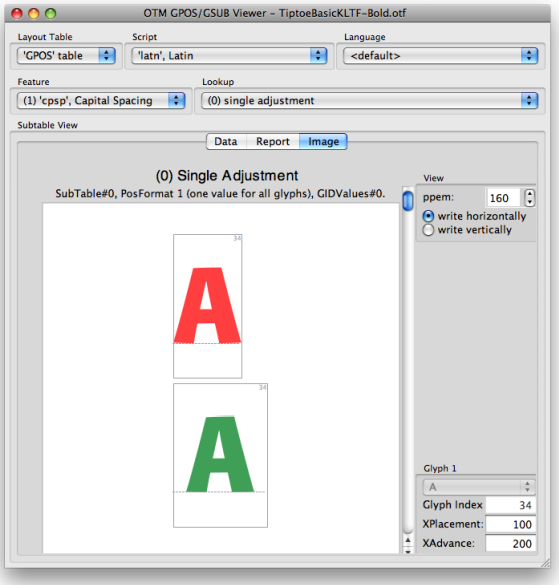
1. **Data**. This is exactly what you see if you inspect 'GSUB' lookups or 'GPOS' lookups in the table content area:



2. **Report**. Provides you with a text report of the selected lookup's content:



3. **Image**. This is the default viewing mode. Substitution or re-positioning is visualized, so that you may evaluate a lookup's layout behavior quickly at a glance, and even evaluate positioning adjustments:



Use the scrollbar to the right of the preview area to flip through all substitutions or positioning adjustments.

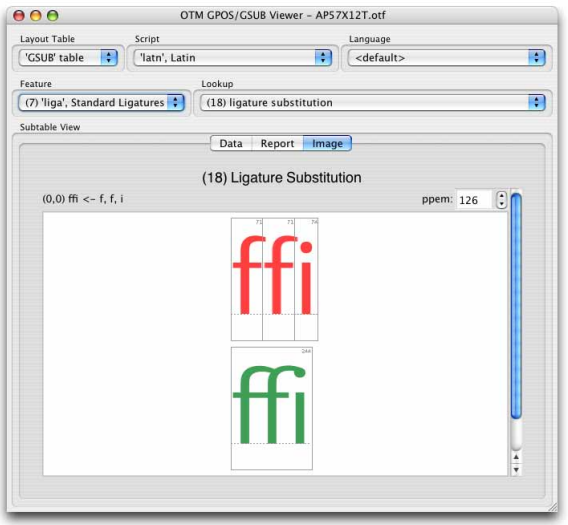
The main area is headed by this lookup's index (in parentheses) and type, followed by additional information like the index of the current subtable or the substitution or positioning format. The original glyph is colored RED, the replaced or repositioned glyph is colored GREEN.

Like in other OTMaster dialogs, use the scrollbar next to the review image to switch from one substitution or positioning entry to another!

Which information are shown in **Image** viewing mode depends on whether you are inspecting 'GSUB' table or 'GPOS' table and also depends on the lookup type. A few examples are given below.

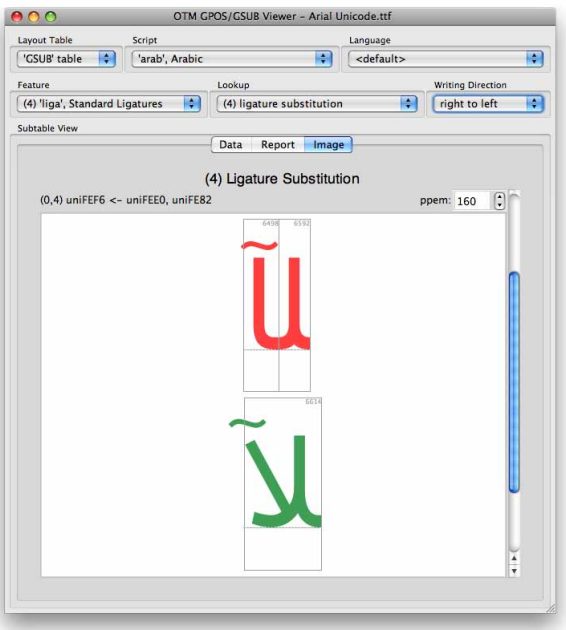
1. GSUB:

1.1. With non-contextual substitution, there are no further options. You may use the scrollbar to flip through glyph substitutions:

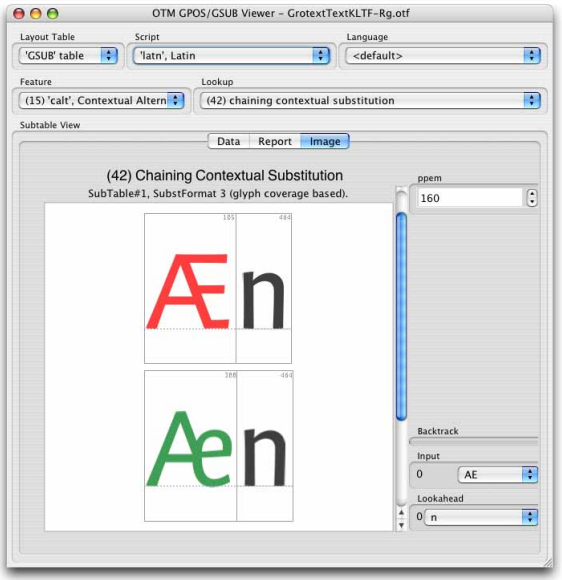


ppem is the ppem size at which the review image is rasterized.

Since version 2.0 of OTMaster, the 'GPOS'/'GSUB' Viewer offers an additional option **Writing Direction**. to change the display order.



1.2. With contextual substitution, there are a few more pieces of information. Besides a RED original glyph and a GREEN adjusted glyph, there is a BLACK context glyph in the preview area:



— ppem

The size at which the review image is rasterized.

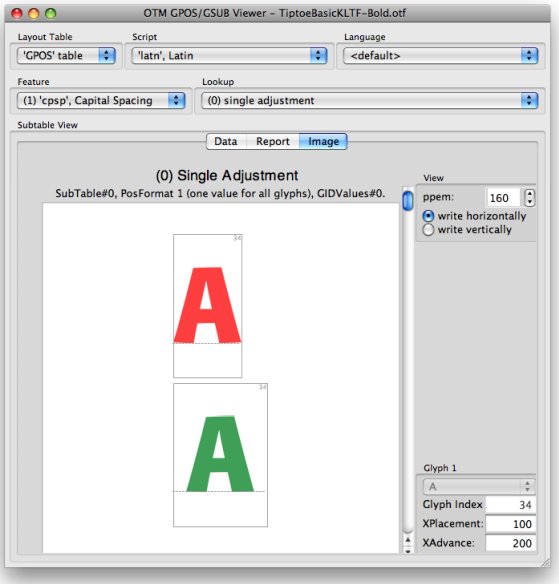
— Backtrack
— Input
— Lookahead

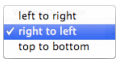
Input is always shown, this is the glyph or glyphs will be substituted, together with **Backtrack** and/or **Lookahead**. Contextual substitution may involve more than just one of each!

If (in the order of appearance) **Backtrack** or **Input** or **Lookahead** involves but a single glyph, the respective popup is greyed out and shows the single glyph's name. If **Backtrack** or **Input** or **Lookahead** involves a glyph class, then the popup shows the first glyph by default. You may use the popup to select any other glyph: just click on the popup and move the mouse up or down the list, and the review will instantly show the glyph touched by the mouse – no need to click.

2. GPOS:

2.1. With single positioning, the affected glyph as well as the positioning adjustment value are shown:





***Writing Direction** options are available in the GPOS too, of course.*

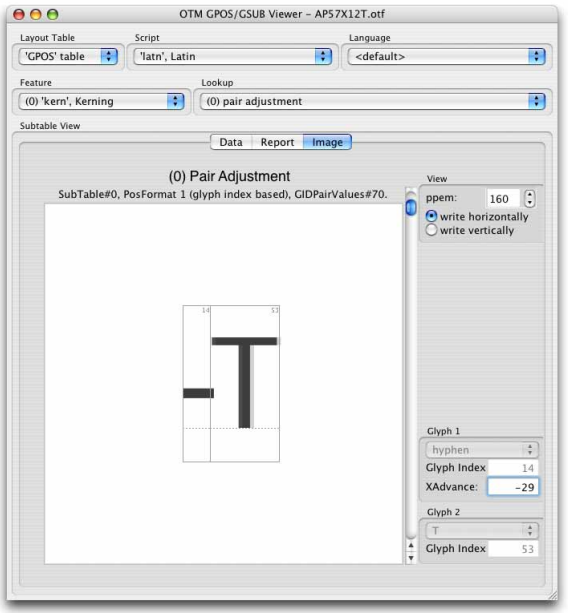
— View

ppem is the ppem size at which the review image is rasterized.
In addition, two radio buttons allow to choose from **write horizontally** and **write vertically** which will arrange the glyphs involved either side by side or one above the other.

— Glyph 1

For each glyph there is a popup with the glyph name. **Glyph Index** is the selected glyph's index. Depending on which kind of adjustments have been made, values for **XPositioning**, **XAdvance**, **YPositioning** and/or **YAdvance** adjustment are shown.

2.2. With pair positioning, there are a few additions:



— View

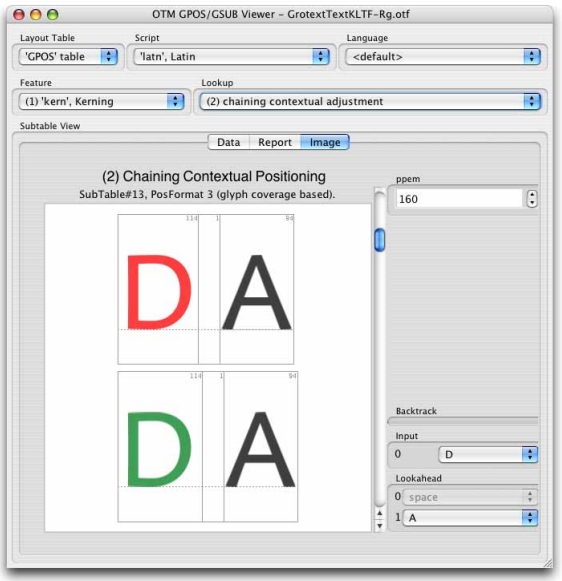
ppem is the ppem size at which the review image is rasterized. Again, two radio buttons allow to choose from **write horizontally** and **write vertically** which will arrange the glyphs involved either side by side or one above the other.

— Glyph 1

— Glyph 2

For each of them there is a popup with the glyph name. If it is but a single glyph, the popup will be greyed out. If there is more than one, you may click on the popup and move the mouse up or down the list to see any of the other glyphs in the review – no need for clicking. **Glyph Index** is the glyph index of the glyph currently shown. Depending on which kind of adjustments have been made, values for **XPositioning**, **XAdvance**, **YPositioning** and/or **YAdvance** adjustment are shown. The latter values can be adjusted. However, the general structure of GSUB and GPOS tables cannot be changed: it is not possible to add anything that has not been in these tables before, nor to remove anything.

2.3. With contextual positioning, BLACK context glyphs join RED original glyph and GREEN adjusted glyph:



— ppem

The size at which the review image is rasterized.

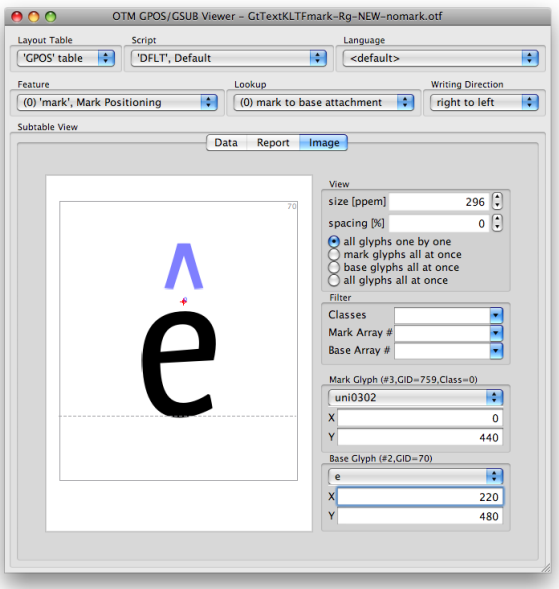
— Backtrack
— Input
— Lookahead

Input is always shown, this is the glyph or glyphs to be substituted, together with **Backtrack** and/or **Lookahead**.

No values are shown, but the effect is visible in the preview. In the example above, the 'D's right sidebearing is increased when followed by space and a class which contains an 'A'.

Note: The 'GSUB'/'GPOS' Viewer allows you to review individual *lookups. If a feature refers to more than one lookups, these lookups' behavior will be additive. This has different effects with* GSUB *and* GPOS.
GSUB*: All lookups associated with a feature will be applied. However, if a previous lookup has substituted an input glyph already, additional lookups will not find a match for the same input glyph any more because this has been substituted already and does not exist any more in the input string. In so far, lookup order matters, as does the order of substitutions inside of a subtable.*
GPOS*: And again, all lookups associated with a feature will be applied. If more than one lookup adjust positioning and/or advance width of a specific glyph, then all lookups' adjustments will* add up. *It is important to keep this in mind, because the 'GSUB'/'GPOS' Viewer visualizes only individual lookups' adjustments, i.e. does not show the result of a feature's total positioning adjustments!*

2.4. Mark-to-base positioning serves to define where mark glyphs (accents as well as vowels and dots in Arabic-script fonts) attach to base glyphs (letters). Each mark glyph carries an anchor point and is associated with a mark class, and each base glyph carries an anchor point per mark class. Thus anchor points determine where on a base glyph a mark glyph sits.



An anchor point is visualized by a red cross. Its position, in relation to the (black) base glyph, can be adjusted via drag/drop of the red cross. Its position, in relation to the mark glyph, can be adjusted via drag/drop of the (blue) mark glyph. Numerical adjustment is possible too.

— View

size [ppem] is the ppem size at which the review image is rasterized. **spacing [%]** is the distance between the glyphs' boxes, in percent of ppem. There are four display options – **all glyphs one by one** (a single mark-to-base pair), **mark glyphs all at once** (a row of all mark-to-base pairs for the selected base glyph), **base glyphs all at once** (a row of all mark-to-base pairs for the selected mark glyph), **all glyphs all at once** (a table of all mark-to-base pairs with a column per base glyph and a row per mark glyph).

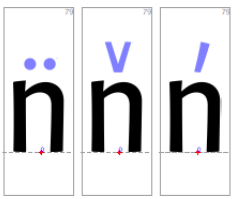
— Mark Glyph

— Base Glyph

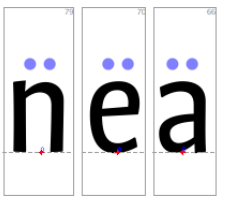
For each mark and base there is a popup from which a glyph can be chosen by name. Which glyphs are shown in each popup depends on the selected **Classes** in the **Filter** section (usually there are separate classes e.g. for top and bottom mark glyphs). **X** and **Y** are the anchor's coordinates.

Below are examples for the four display options.

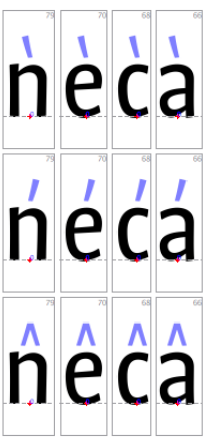
all glyphs one by one



mark glyphs all at once



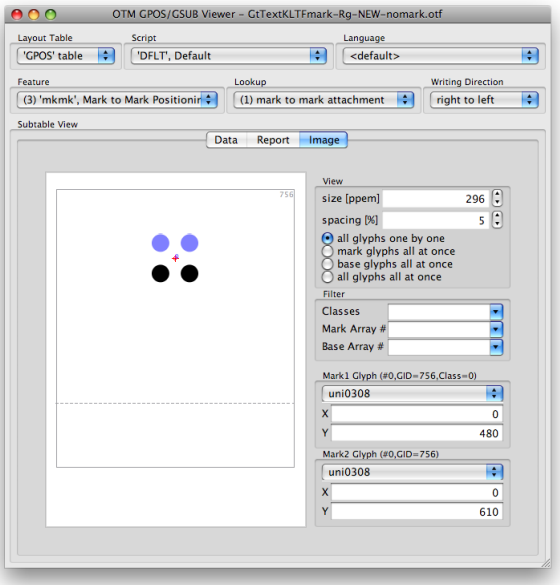
base glyphs all at once



all glyphs all at once

Note: *Adjusting a mark glyph's anchor position will reposition this mark for* all *base glyphs. And adjusting a base glyph's anchor position will reposition* all *marks associated with this anchor.*

2.5. Mark-to-mark positioning serves to define where mark glyphs attach to other mark glyphs. Each mark glyph (attaching to another mark glyph) carries an anchor point and is associated with a mark class, and each mark glyph (allowing another mark to attach to it) carries an anchor point per mark class.



The interfaces for mark-to-mark positioning and for mark-to-base positioning are the same.

This resembles mark-to-base positioning. As a consequence, the dialog and editing behavior is identical.

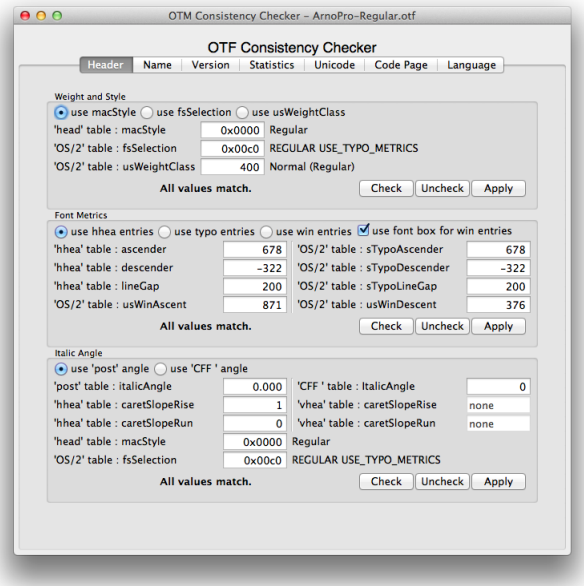
Consistency Checker

This tool helps finding inconsistencies across OpenType fonts' tables. Most Consistency Checker dialogs follow the same pattern. They present table entries which are expected to be consistent. **Check** compares entries' values and suggests corrections. **Uncheck** reverts to original values. And **Apply** finally applies automatically or manually refined values to the font.

Tip: In addition, consider using font checking tools like Microsoft's Font Validator ►www as well as the command-line tool CompareFamily ►www which is included in Adobe's AFDKO.

— Header

The Header section of the Consistency Checker gives an overview, in one dialog, of different table's entries which need to be consistent:



1. **Weight and Style** compares the OS/2 table's **fsSelection**, **usWeightClass** and the head table's **macStyle**. If a font is a bold style, this should be reflected in all three entries, if a font is an italic style, this should be reflected in **fsSelection** and **macStyle**.

Please see the OS/2 table spec ►*local* ►*www* for **fsSelection** and **usWeightClass** with the head table spec ►*local* ►*www* for **macStyle**.

2. **Font Metrics** compares the hhea table's **ascender/descender** (the **hhea entries**) with the OS/2 table's **sTypoAscender/sTypoDescender** and **usWinAscent/usWinDescent** (the **typo entries** and **win entries**). Via the top row radio buttons you may choose against which of these entries the other ones will be checked. The **use font box for win entries** option derives the OS/2 table's **usWinAscent/usWinDescent** values from the largest glyph dimensions found in the font.

Please review **Check**'s adjustments – you may not agree with them because it sets:

ascender = sTypoAscender = usWinAscent
descender = sTypoDescender = usWinDescent

Both UPM and the three sets of vertical metrics

a. head **unitsPerEm**
b. hhea **ascender/descender/lineGap**
c. OS/2 **sTypoAscender/sTypoDescender/sTypoLineGap**
d. OS/2 **usWinAscent/usWinDescent**

need to be defined carefully because different applications pick different entries to determine the default line-to-line distance. To achieve consistent default line-to-line distance in most –though not in all– applications, you need to take care that the 'sum' of each of the first three sets is the same. Microsoft's and Adobe's current practice for determining (vertical) font metrics –which may be interpreted as a recommendation– is this:

2.1 The pair **sTypoAscender/sTypoDescender** indicates how much of UPM (the head table's **unitsPerEm**) is reserved for ascenders and descenders. This can be expressed as:

sTypoAscender – sTypoDescender = UPM

It is expected that your typeface is designed and UPM is defined such that normal ascenders and descenders as of 'A'–'Z' and 'a'–'z' remain within the UPM's boundaries, or sTypoAscender and sTypoDescender.

2.2 **sTypoAscender/sTypoDescender/sTypoLineGap** (set c) defines a font's ideal line-to-line distance. It depends on the typeface's design, but usually **sTypoLineGap** is about 20% of:

2.2.1 sTypoAscender – sTypoDescender
2.2.2 UPM

(According to 2.1, 2.2.1 and 2.2.2 are equal.) For example, UPM = 1000 and sTypoLineGap = 200 result in a default line-to-line distance of 120% of UPM which would translate into 10/12 pt.

2.3 Since **sTypoAscender/sTypoDescender/sTypoLineGap** (c) were defined such that the 'sum' results in an ideal default line-to-line distance, you may set the OS/2 table's version to 4 and **fsSelection** bit 7 to 1. This indicates that **sTypo**-values should be used for calculating default line-to-line distance rather than **usWin**-values. Once the OS/2 table's version is 4 however, you need to set **fsSelection** bits 8 and 9 consciously!

2.5 **sTypoAscender/sTypoDescender/sTypoLineGap** (c) and **ascender/descender/lineGap** (b) share same values, so you can simply reuse the first set's values for the latter set:

ascender = sTypoAscender
descender = sTypoDescender
lineGap = sTypoLineGap

2.6 **usWinAscent/usWinDescent** (d) provide information about a font's clipping zones – the largest extensions that you do not want to see clipped, i.e. cut off. These values reflect the tallest glyphs in the font.

An example that meets all conditions:

head			
UPM			= *1000*
OS/2	hhea		
sTypoAscender	= *ascender*	= *800*	
sTypoDescender	= *descender*	= *-200*	
sTypoLineGap	= *lineGap*	= *200*	
usWinAscent		= *850*	
usWinDescent		= *350*	

Tip: *Please consult John Hudson's* Setting Cross-Platform Vertical Metrics, *especially the latest 'Update'.* ▶ ***www*** *Rather for the illustration than the method itself which is obsolete now, see Karsten Lücke's* Font Metrics. ▶ ***www***

Note: *This is a recommendation, not a requirement! Also see* ▶ OS/2.

Note: *The* hhea *table's* ***descender*** *&* ***the*** OS/2 *table's* ***sTypoDescender*** *are given as negative values – but* ***usWinDescent*** *must be given as a positive value! Therefore, the term 'sum' ist not exact in a strict sense.*

With a typeface of normally sized ascenders and descenders, the 'sum' of the **usWin** set will be smaller than the 'sum' of the **sTypo** set. Increase **usWin**-values a bit so that the 'sum' of each of the three sets is identical:

sTypoAscender – sTypoDescender + sTypoLineGap (c)
= usWinAscent + usWinDescent (d)
= ascender – descender + lineGap (b)

In case that a font contains excessively tall glyphs it may be impossible to achieve this equation.

It is strongly recommended that you define vertical metrics such that they are identical across all fonts that belong to a family. This to make sure that if a user relies on default line-to-line distance, this would not vary if one paragraph is set in Regular, the other in Italic or Bold style.

3. **Italic Angle** allows the comparison of the post table's **italicAngle** with the OS/2 table's **ItalicAngle** and the hhea table's **caretSlopeRise** and **caretSlopeRun**. The relation between these two is

tan –italicAngle = slopeRun / slopeRise

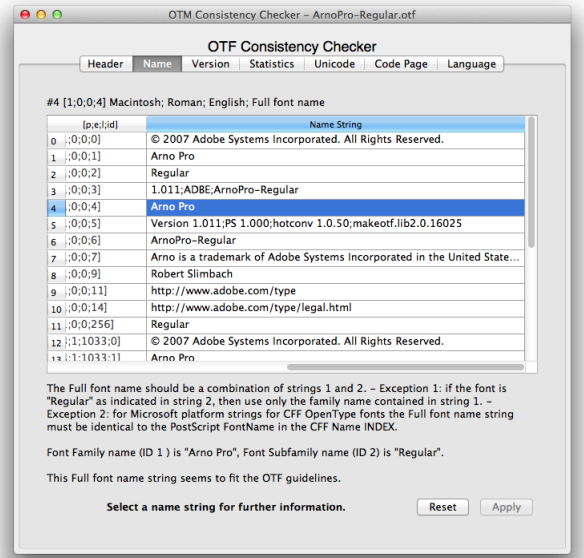
The vhea table's **caretSlopeRise** and **caretSlopeRun** can be checked too if this table is present.

For the sake of consistency, the OS/2 table's **fsSelection** and **macStyle** are shown again, since their values need to be in tune with italic angle and slope rise and run.

*Tip: If your font is a CFF-based OpenType font, also compare with the 'CFF' top dictionary **ItalicAngle**, and possibly with slanting as results from the **FontMatrix** which is located in 'CFF' top dictionary as a nested table.*

— Name

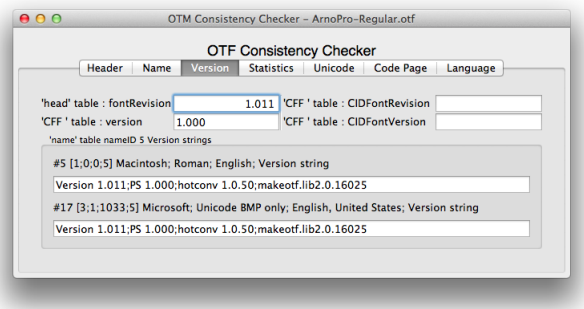
The Name section of the Consistency Checker helps inspecting individual name table entries and serves as a built-in documentation of the name table nameIDs, pointing out their correlation with other table's entries. So you may flip through your name table's records and compare your entries with the recommendations.



Reviewing the name record with NameID *2 (the* Subfamily *name, Macintosh platform).*

— Version

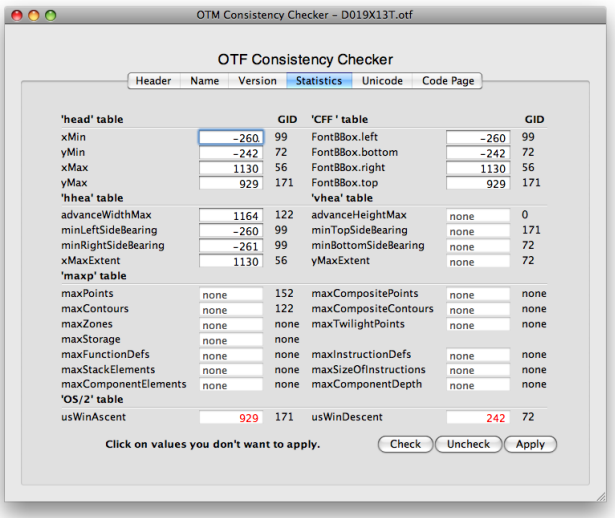
The Version section displays, side by side, various version information as found in an OpenType font.



A font's version information side by side.

— Statistics

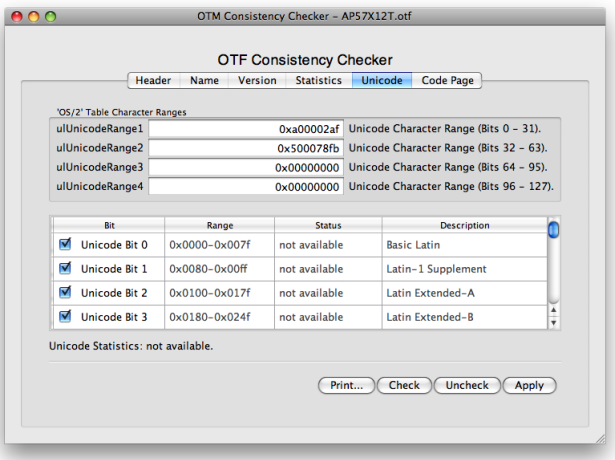
Upon clicking the **Check** button, the Statistics section of the Consistency Checker will gather minimum and maximum metrics. It also shows the glyphs to which these metrics relate (identified by **GID**). This makes it easy to track down possibly problematic glyphs. Values which Consistency Checker considers to be wrong are colored in red.



Various metrics and the glyphs to which they relate. The value 'none' indicates that either the according table does not exist or that the table version does not include these data.

— Unicode Ranges

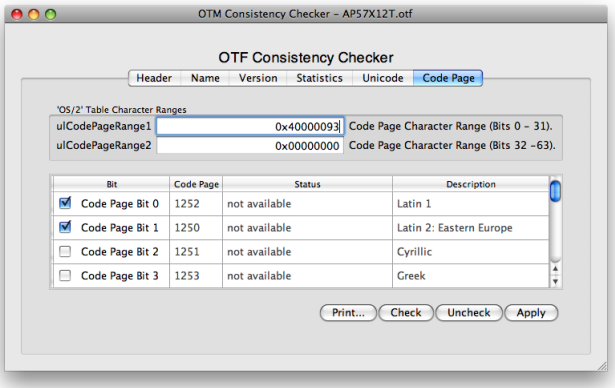
The Unicode Ranges section of the Consistency Checker shows the OS/2 table **ulUnicodeRange1**–**ulUnicodeRange4** values and presents individual bits and their meaning in a list. Editing behavior is described in context of the ▶ *Codepage Ranges* section.



***ulUnicodeRange** bytes represented in hexadecimal form and as individual bits.*

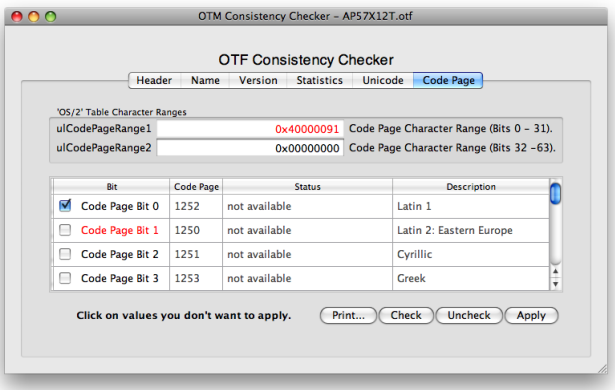
— Codepage Ranges

The Codepage Ranges section of the Consistency Checker shows the OS/2 table **ulCodePageRange1** and **ulCodePageRange2** values and presents individual bits and their meaning in a list.



***ulCodePageRange** bytes in hexadecimal form and as individual bits.*

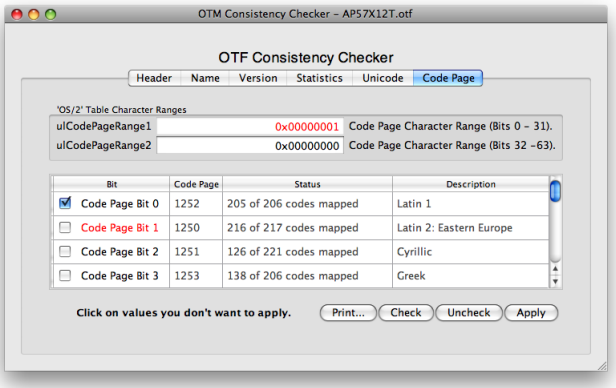
Clicking a checkbox will change the state of the according bit and highlight the adjustment in red – including in the hexadecimal representation:



Your adjustments highlighted in red.

Accept adjustments by clicking the **Apply** button.

Upon clicking the **Check** button, the Codepage Ranges section (like the Unicode Ranges section) will show, in the **Status** column, how many of the characters referenced by each Codepage Range (or Unicode Range) are covered in the font's cmap table:

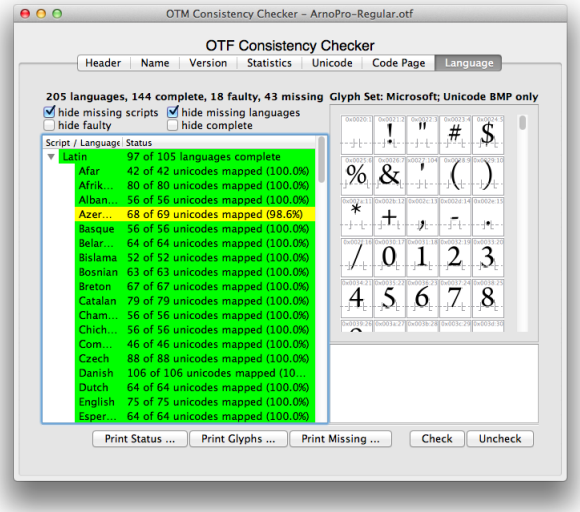


***Check** how many of a Codepage Range's characters are present in a font and activate the relevant bits.*

This also deactivates all bits except those that are relevant based on the actual character coverage. Again click **Apply** to accept the changes.

— Languages

The Languages section of the Consistency Checker compares the font's coverage of Unicode codepoints against the ICU database. It reports which scripts and languages are completely covered (green), incomplete (yellow) or are entirely missing (red). Scripts are first in hierarchy, languages are second and are shown after clicking the triangle in front of a script's name.



A character is missing for Azerbaijani.

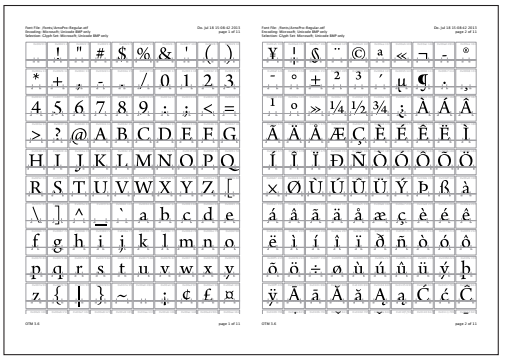
To avoid getting lost in irrelevant information, you may **hide missing scripts**, **hide missing languages**, but also **hide incomplete** or **hide complete** entires.

If you select an incompletely covered language, you are shown a list of missing characters in the bottom right area:

This character is missing!

There are three printing options. **Print Status ...** reflects the window's leftside area, the script/language overview:

Print Glyphs ... reflects the window's rightside area, the glyph overview:



Print Missing ... is a list of missing glyphs as of the bottom right area:

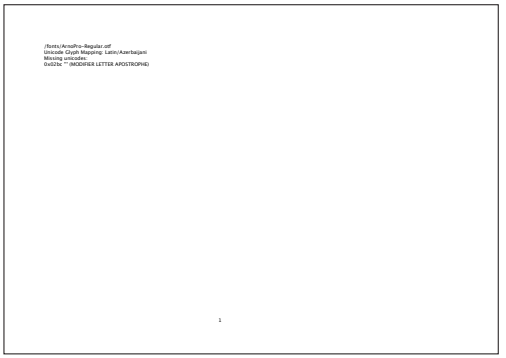


Table Comparator

This tool displays the content of fixed-length tables side by side. Currently the tables head, hhea, vhea, OS/2, maxp, post, CFF and PCLT are supported. The leftmost column is reserved for the font which is selected in OTMaster's main dialog. Other fonts are compared against it. Identical values are marked in green, differences are marked in red.

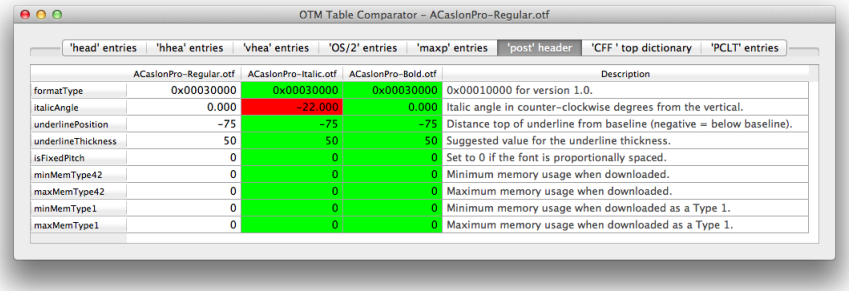
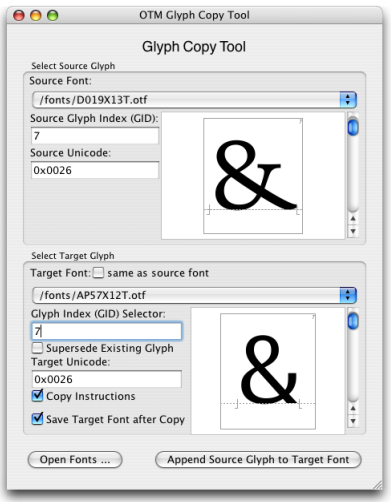


Table values can be edited right in the Table Comparator.

Glyph Copy Tool

This tool makes it easy to copy a glyph from one font to another, or to duplicate a glyph within the same font e.g. to add a same-looking but differently named and reencoded character.



The Glyph Copy Tool's dialog.

***Tip:** Use the Glyph Copy Tool to copy outlines from CFF-based to TT-based OpenType fonts and vice versa. It will automatically convert outlines into the required format and scale them according to the destination font's UPM as defined in the head table.*

— Select Source Glyph

Source Font offers all fonts opened in OTMaster in a popup menu. To copy a glyph from a font not opened yet, use **Open Fonts …** to add it to popup list.
Source Glyph Index (GID) is the index of the glyph you want to copy.
Source Unicode is the Unicode codepoint of the glyph you want to copy.
So you may determine the source glyph either by its index, its Unicode codepoint, or by using the scrollbar next to the glyph image to flip through all glyphs and choose by appearance.

— Select Target Glyph

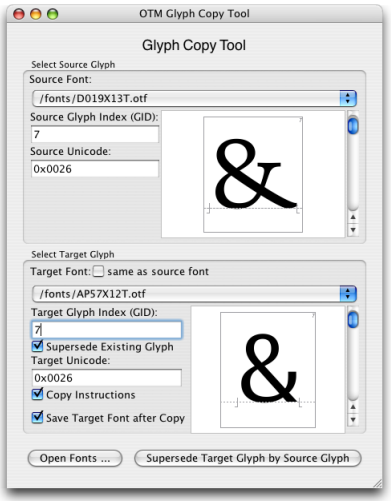
Target Font offers all fonts opened in OTMaster in a popup menu.
In case that you intend to merely duplicate a glyph within the same font, activate the checkbox **same as source font**.
Glyph Index (GID) Selector does not have a function as long as you do not intend to supersede (i.e. replace) an existing glyph. Glyphs always are appended as the last glyph of the font.
Supersede Existing Glyph – rather than appending the new glyph to the glyph set, activating this option will replace an existing glyph which is defined by the **Target Glyph Index (GID)** right above the checkbox.
Target Unicode is the Unicode codepoint for the new glyph.
Copy Instructions will copy hinting information with CFF-based OpenType fonts.

***Note:** When copying a glyph, the newly appended glyph's name is derived from its destination glyph index. To give it a friendlier name, you may want to open the Glyph Viewer and change the 'CFF' id or 'post' name.*

***Note:** Hinting instructions are ignored with TT-based fonts.*

Save Target Font after Copy will save the target font immediately after appending or replacing a glyph. This is necessary for OTMaster to be able to rasterize the newly appended e.g. in the Font Viewer or Glyph Viewer.

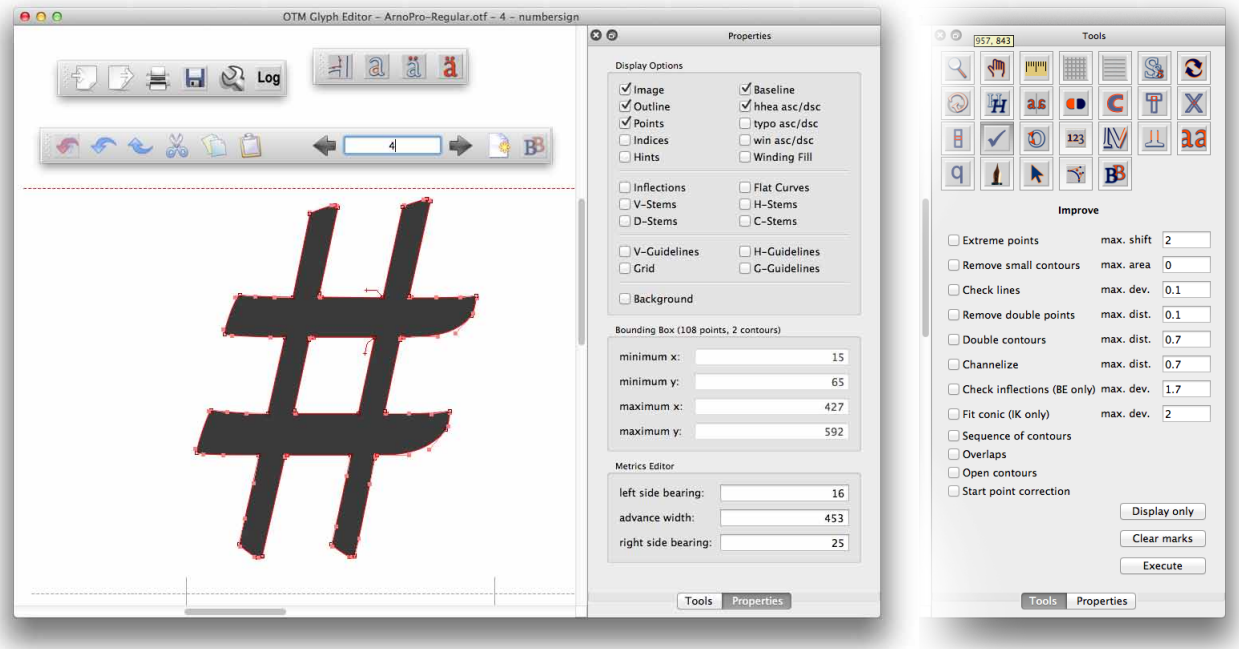
The above image's settings are for appending a font's glyph to another font. As another scenario, the settings in the image below are such that a source glyph will supersede (i.e. replace) the 'same' glyph of another font – here we replace Argo's ampersand by Documenta's:



*Tip: For this reason, it is highly recommended that you only edit copies of fonts with OTMaster, to prevent that you inadvertedly overwrite a font when using **Save** while editing. Either, make a copy of the font file, and edit the copy in OTMaster. Or, **Save As…** a font immediately after opening it, by another name.*

Glyph Editor

OTMaster is not intended to be a full font editor (font editor in the sense in which FontMaster or its components BezierMaster or IkarusMaster are). But we did not want to miss some basic glyph editing functionality even in a 'mere' table editor as OTMaster. What if you find duplicate contours in a glyph of a font ready to be distributed? Go back to the font editor of your choice, open the font's source data, correct the error, generate the font again, possibly repeat additional production steps with AFDKO or VOLT or at least perform some tricks to smuggle new outline data into the previous font version? Glyph Editor helps.



The Glyph Editor's toolbars can be rearranged by simple drag & drop. They can be placed anywhere on the screen, like in the screenshot above. They can also be attached to any of the Glyph Editor window's left or right edges. The dock widget to the right shows either various display **Properties** or an overview of editing **Tools** and the options of the one currently selected.

The Glyph Editor window consists of two areas: a main area for glyph editing and a dock widget presenting the tools.

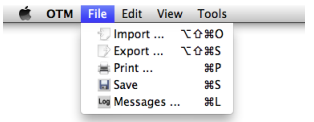
*There are three toolbars: **File** for importing, exporting and printing glyphs, the usual **Edit** functions including a textbox for selecting a glyph by glyph index, and various **Selection** modes. **View** modes and **Tools** are located in the dock widget which is new in OTMaster 3.6.*

Defaults can be defined in the ▶Preferences.

The Glyph Editor has three toolbars whose functions are reflected in menus too. Most of the functions can be accessed by way of shortcuts.

FILE

This menu and toolbar allows to import, export and print glyphs:



Import ...

Imports individual glyphs. Supported formats are:

— **EPS Encapsulated PostScript** [.eps]
— **SVG Scalable Vector Graphics** [.svg]

Export ...

Exports individual glyphs. OTMaster can export:

— **EPS Encapsulated PostScript** [.eps]
— **SVG Scalable Vector Graphics** [.svg]
— **Editable SVG Scalable Vector Graphics** [.svg]

The difference between the two SVG versions is that the former is an SVG font while the latter is an SVG illustration.

Print ...

Prints the current glyph.

Save ...

Save changes.

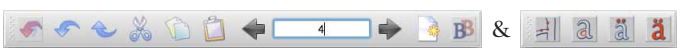
Character Preferences

Opens the Preferences dialog e.g. changing colors of outlines, points and guidelines. Please see the ▶ *Preferences* chapter.

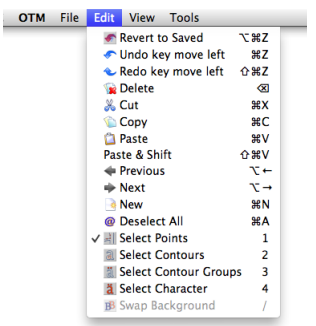
Messages

Opens the Messages window which displays all status messages.

EDIT & SELECTION



The edit toolbar holds the basic editing functions like **Copy** and **Paste**, but also **Undo** and **Redo**, and arrows to flip through glyphs.
The selection toolbar is for changing the selection mode.
The edit menu (see the righthand image) holds both the editing functions and selection modes.



Revert to Saved

Un-does all changes since the font has been saved.


Undo (⌘+Z) (CTRL+Z)
Un-does previous changes.


Redo (⌘+Y) (CTRL+Y)
Re-does un-done previous changes.


Delete (⌫)
Deletes the selected points or contours.


Cut (⌘+X) (CTRL+X)
Cuts the selected points or contours. This will remove them from the glyph and keep them in the clipboard.


Copy (⌘+C) (CTRL+C)
Copies selected points or contours into the clipboard without removing them from the glyph.


Paste (⌘+V) (CTRL+V)
Pastes points or contours from the clipboard into the glyph, or pastes a table entry from the clipboard into the currently selected table.

Paste & Shift
Pastes points or contours into the glyph, but with a slight offset.

Previous (⌘+P) (CTRL+P)
Next (⌘+N) (CTRL+N)
Goes to the previous or next glyph. You may also enter the glyph index (GID) into the textbox – and do not forget to confirm with ↵!


New (⌘++) (CTRL++)
Closes all open fonts. If you have made any changes which have not been saved yet, you will be asked whether to save changed fonts or not.

Select All / Deselect All (⌘+A) (CTRL+A)
Selects all of the glyph's points or contours. If all of a glyph's points or contours are selected already, deselects all of them.


Select Points
Select points via mouse. Add points to (or remove them from) the selection by holding the ⇧ key and clicking on unselected (or selected) points.


Select Contours
Select individual contours by clicking inside of a contour. Add contours to (or remove them from) the selection by holding the ⇧ key and clicking on unselected (or selected) contours.

Select Contour Groups

Select contour groups by clicking inside of a contour. Unlike the previous selection mode, Select Contour Groups will select not only the contour you have clicked on but also all contours inside of it. Add contour groups to (or remove them from) the selection by holding the ⇧ key and clicking on unselected (or selected) contours.

Select Character

Select the entire character by clicking inside of any contour.

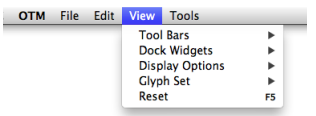
Swap Background

Swaps foreground glyph and background glyph.

The following menus correspond to the **Properties** and **Tools** dock widgets rather than to toolbars.

VIEW

The view menu serves to define various view options.



Tool Bars

Check or uncheck this submenu's items to show or hide the **File**, **Edit** and **Selection** toolbars (which have been described above).

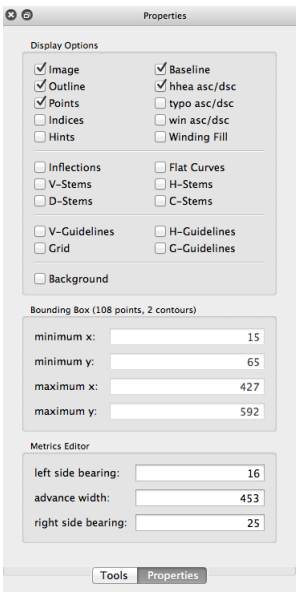
Dock Widgets

Check or uncheck this submenu's items to define which of the two widgets, editing **Tools** and display **Properties**, are available in the Glyph Editor.

Display Options

This menu's options essentially correspond to those in the **Properties** dock widget which is shown here.

***Display Options** as of the **Properties** dock widget.*



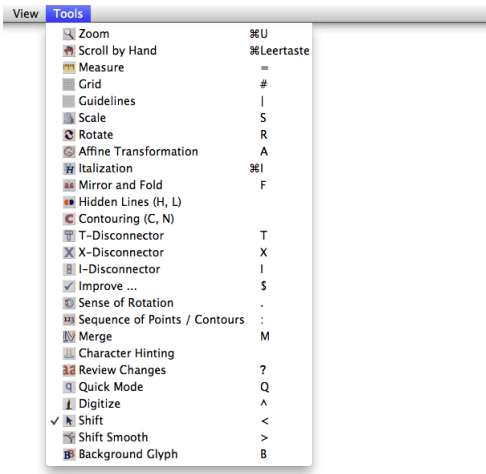
Glyph Set

Here you may choose the glyph set which you would like to access in the Glyph Editor: **Glyph Index (GID)**, **Unicode 2+ semantics BMP only**, **Macintosh Roman**, or **Microsoft Unicode BMP only**.

Reset (F5)

This will restore the default glyph size.

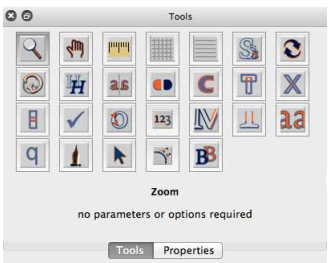
TOOLS



And finally, the tools menu and dock widget offer functions for editing and designing glyphs.

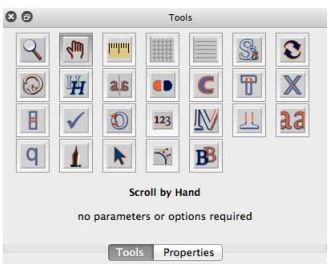
Zoom

Click into the editing area to zoom in. Hold down the ⇧ key, then click into the editing area to zoom out. The position of the mouse will define the center of the new view. Alternatively, hold down the mouse button and draw a rectangle to zoom into this segment of the editing area.



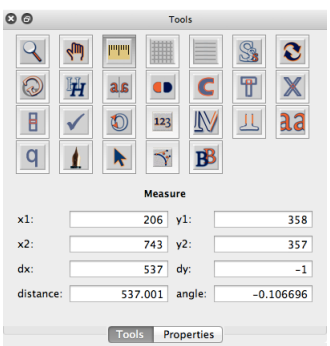
Scroll by Hand

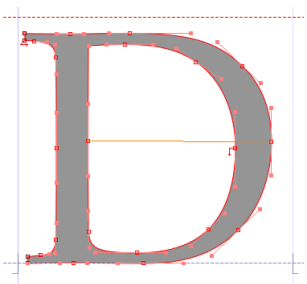
Click anywhere into the editing area, hold the mouse button down and move the mouse around to scroll the entire editing area.



Measure

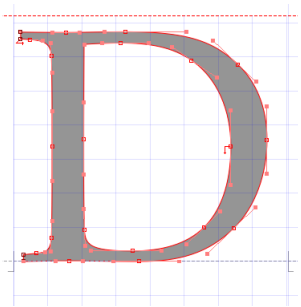
Click on a start point, move the mouse while holding down the mouse button, and release the button at the end point. The dialog will show information about the start point's **x1** and **y1**, the end point's **x2** and **y2**, the x- and y-distances between both points as **dx** and **dy**, the **distance** between both points, and the **angle**.

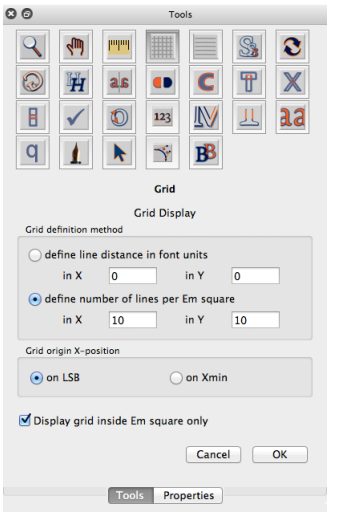




Grid

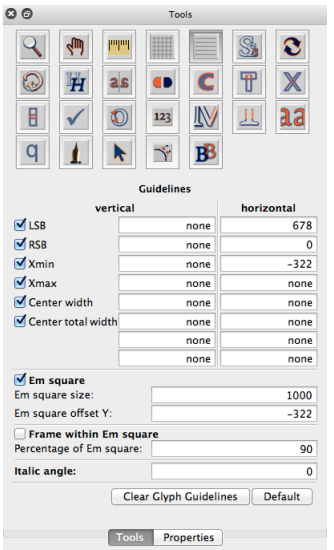
Display a grid in the background. You may define the line-to-line distance in font units or as lines per EM square, and decide at which origin point the grid is meant to start.

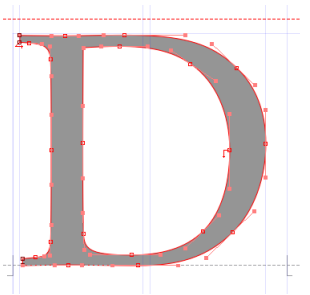


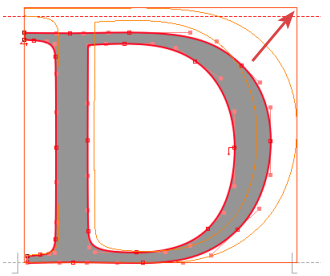


Guidelines

Here you may define a variety of horizontal and vertical guidelines which may be displayed in the background.

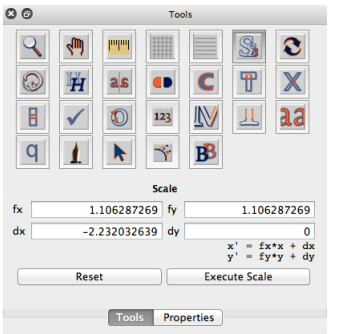






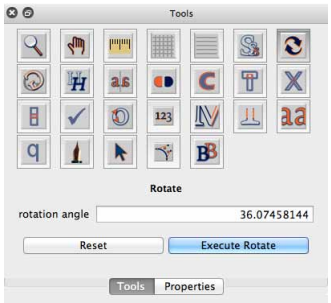
Scale

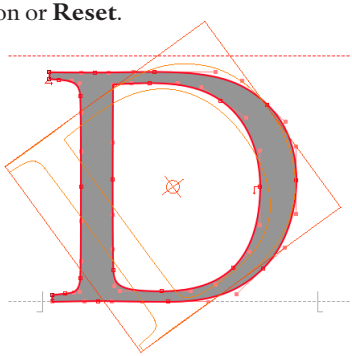
Scale the glyph by mouse or by entering precise scaling factors **fx** and **fy** and position adjustments **dx** and **dy**.



Rotate

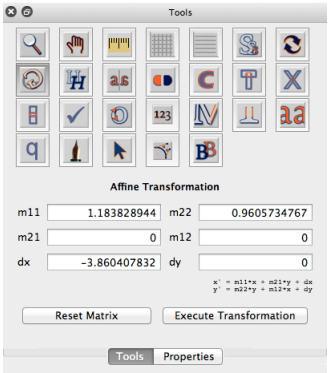
Use the mouse to rotate the glyph in the edit area, or enter a **rotation angle** in the editing area (a negative value is for clockwise rotation) and then **Execute Rotation** to apply the rotation or **Reset**.

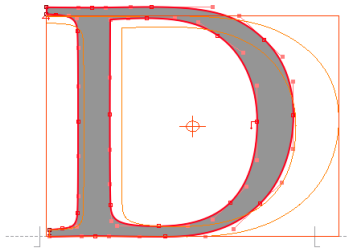




Affine Transformation

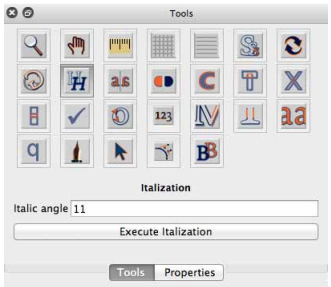
This will transform the glyph in PostScript transformation matrix fashion. The equations are given at the bottom right corner of the editing area:

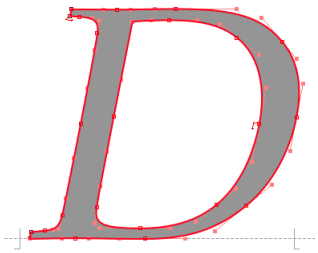




Italization

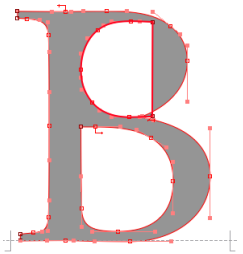
This will italizise, or slant, the selected points or contours, either by using the mouse or by entering a value in the editing area (a positive value will slant to the right side) and confirming with **Execute Italization**.

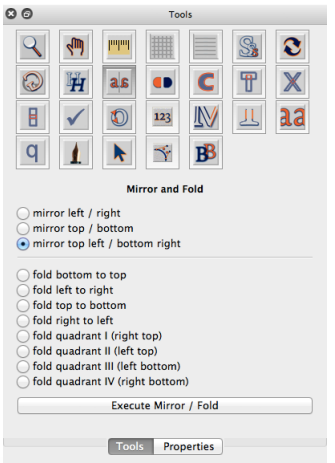




Mirror and Fold

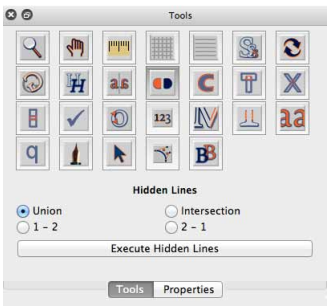
Mirrors a glyph horizontally (**mirror left/right**), vertically (**mirror top/bottom**) or both ways (**mirror top left/bottom right**), and in various additional ways too.

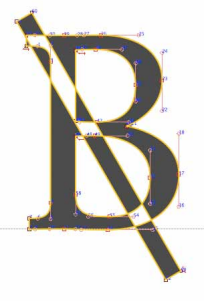




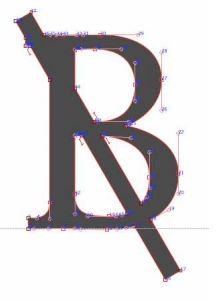
Hidden Lines

This function will remove contour overlaps. The four different way of doing so are illustrated below.

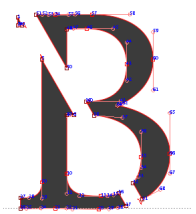




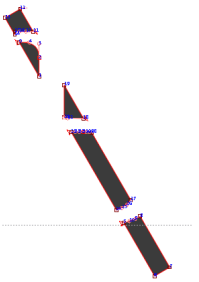
Before removing overlaps with Execute Hidden Lines.



*After removing overlaps with **Union**.*



*With **1–2**: 'B' minus 'backslash'.*



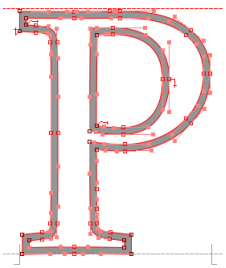
*With **2–1**: 'backslash' minus 'B'.*

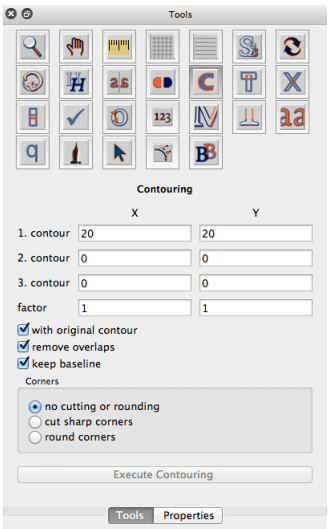
*With **Intersection**.*

Union will logically add overlapping contours. **Intersection** will keep the overlapping (or intersecting) parts. **1–2** will subtract contour 1 from contour 2. **2–1** will subtract contour 2 from contour 1.

Contouring

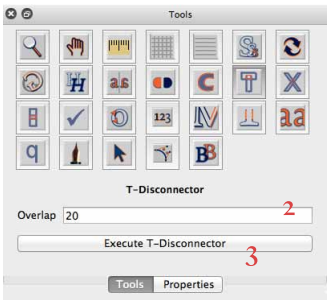
Use Contouring to add up to three contours around selected contours. Determine the distance of each additional **contour** from the original outline, independently for **X** and **Y** direction. Distances are given in units relative to UPM (the head table's **unitsPerEm**).



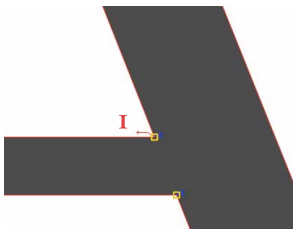


T-Disconnector

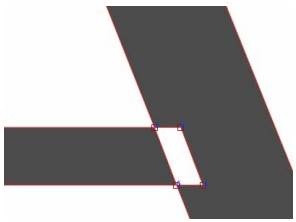
The T-Disconnector will disconnect two parts of a glyph's shapes between any two points, in three simple steps:



To disconnect e.g. the bar from the rightside diagonal of an uppercase 'A',

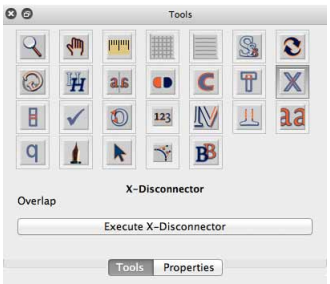


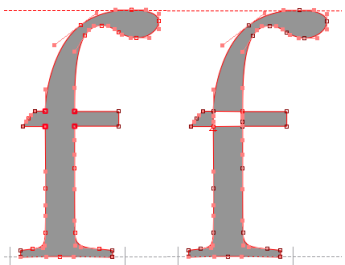
1. select two points at which you want to 'break up' the glyph,
*2. determine the **Overlap** amount in units (relative to the font's UPM) in the T-Disconnector transformation area,*
*3. click **Execute T-Disconnector** to apply disconnection:*



X-Disconnector

This works like the T-Disconnector.



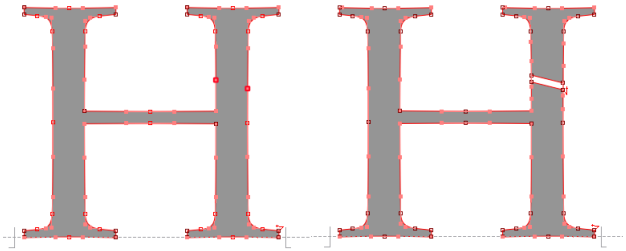


Select points where stems, bars or diagonals cross.

After applying X-Disconnector, crossing stems, bars or diagonals will be disconnected.

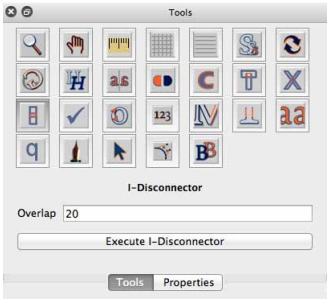
I-Disconnector

The I-Disconnector will disconnect a stem, bar or diagonal at any two selected points.



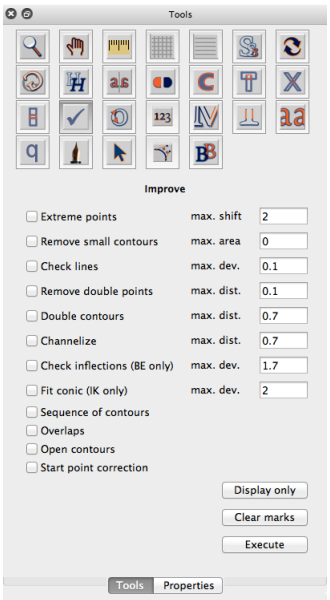
Select two points on a stem or bar where you would like to break it. (You may need to insert points for this purpose.)

After applying I-Disconnector, the stem or bar will be split into two.



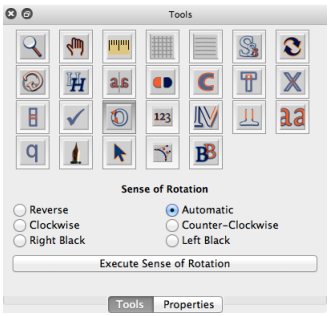
Improve

Outlines are not always flawless from a technical point of view. The image to the right shows aspects that OTMaster can improve. You may merely review OTMaster's finding with **Display only** (and then **Clear marks** again) or apply suggested improvements with **Execute**.



Sense of Rotation

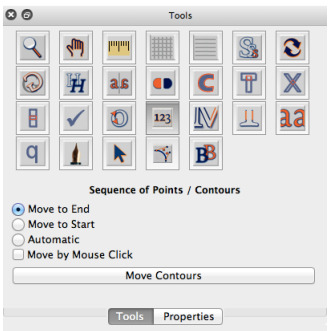
The easiest way to correct contour directions is to select all contours and choose the option **Automatic**. However, you may also select one or more contours (the **Select Contours** mode is recommended for this) and then **Reverse** the current rotation, or define rotation to be **Clockwise** or **Counter-Clockwise**, or by categories **Right Black** or **Left Black**.



Note: In CFF-based OpenType fonts, the outermost contour is expected to be counter-clockwise. In TT-based OpenType fonts, the outermost contour is expected to be clockwise. With both formats though, each contour needs to have the opposite direction than the contour in which it is located.

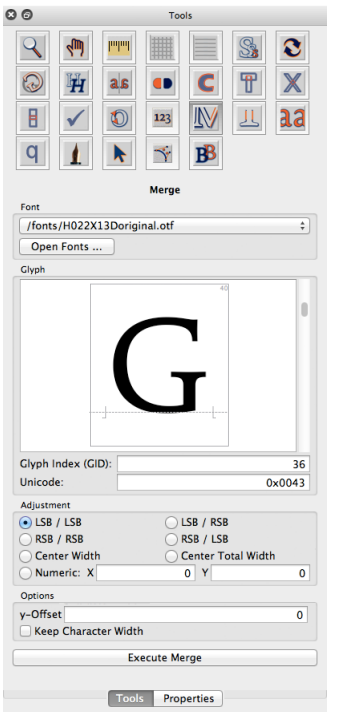
Sequence of Points and Contours

After selecting points or contours, you may change their order by selecting one of the following options:



Merge

To merge one glyph with another one, go to the destination glyph and choose the Merge tool:



— Font

This is the font from which to take a glyph for merging with the destination glyph. You may open a source font with **Open Fonts …**

— Glyph

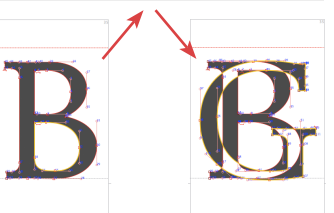
This is the glyph to be merged with the destination glyph. You may select it by using the scrollbar to the right of the preview area, or enter either a **Glyph Index (GID)** or **Unicode** codepoint and confirm with ↵.

— Adjustment

Here you may define how the added glyph shall align with the destination glyph: **LSB/LSB** will position it such that both glyphs' origin points share same coordinates, **RSB/RSB** will align glyphs at the right side, **LSB/RSB** will place the added glyph to the left of the destination glyph, **RSB/LSB** will place the added glyph to the right of the destination glyph, **Center Width** will center-align them (center being calculated from glyphs' outlines, excluding sidebearings), and **Center Total Width** will center-align them (center being calculated from glyphs' total widths, including sidebearings). Or determine the added glyph's position as **X** and **Y** adjustment from the destination glyph's origin.

— Options

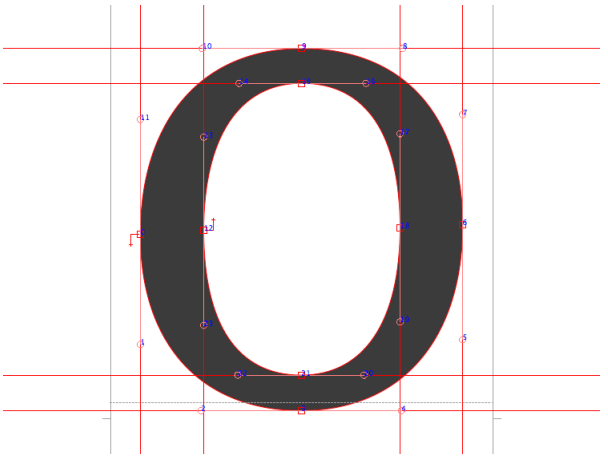
An additional **Y-Offset**.
Keep Character Width of the destination character.



Time to remove contour overlaps with Hidden Lines!

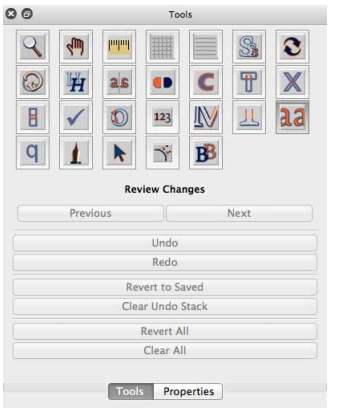
Character Hinting

Character Hinting expresses three choices by way of two buttons: Keep a glyph's original hinting information – this is the default. Click **Omit Original Hints** to remove existing hinting information. Or click the first button to toggle between **Autohinting On** and **Autohinting Off**.



There are some more autohinting options in the Preferences – the kind of stems which the autohinter shall recognize and tolerances, and the glyphs from which alignment zones shall be derived. Please see the ▶ *Preferences* chapter for details.

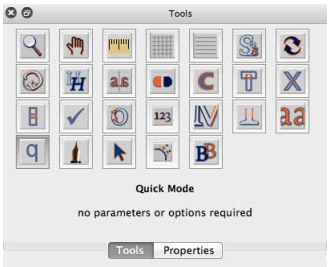
Review Changes



Review Changes presents a history of glyph transformations. Here you may **Undo** or **Redo** individual transformations. Or, per glyph, you either go back to the original version with **Revert to Saved** or accept your changes and **Clear Undo Stack**. Or, for all glyphs, you either go back to their original versions with **Revert All** or accept transformations and **Clear All** undo information.

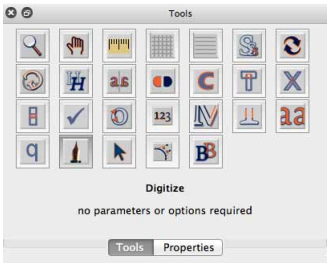
Quick Mode

Choose this mode to select and shift selected points or contours.



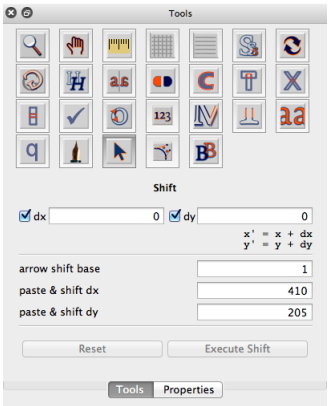
Digitize

Time to reanimate your Aristo tablet!



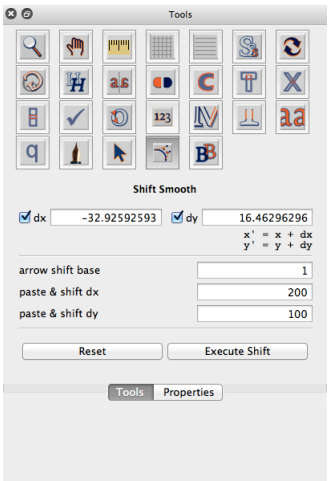
Shift

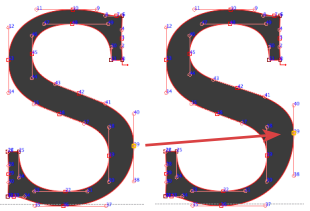
Shift by mouse, at **arrow shift base** units, or shift numerically, at **dx** and **dy** units. Additional **paste & shift dx** and **past & shift dy** parameters allow you to define an offset at which contours are pasted, relative to their original positions.



Shift Smooth

Shifting a point around will interpolate points between the selected point and neighbor extremum points, making sure that curves remain smooth.

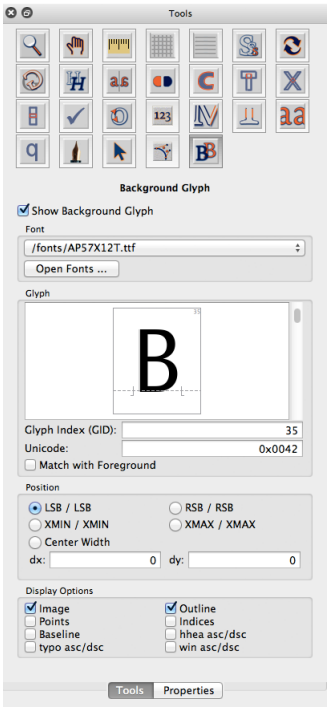




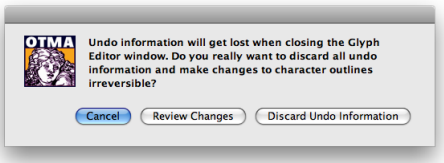
In Shift Smooth mode, surrounding curves (up until neighboring extremum points) will remain smooth when moving a point.

Background Glyph

This tool allows you to select a background glyph. First, select one of the fonts opened in OTMaster or open another font. Then, select a glyph by **Glyph Index** or **Unicode** codepoint. Finally, determine the position of the background glyph relative to the foreground glyph, and which of its information, e.g. **Points**, **Outlines**, **Indices**, you would like to see.



Closing the Glyph Editor window will produce the following dialog:



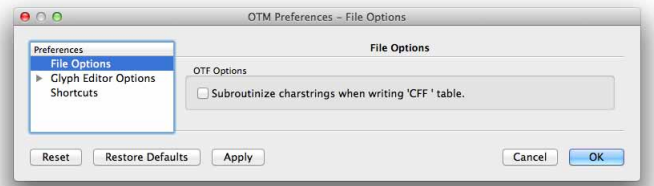
Cancel will get you back to the Glyph Editor so that you may continue drawing.
Review Changes will get you back to the Glyph Editor too and present Undo and redo functions – allowing you to reconsider whether or not you want to apply individual transformations. See ▶ *Review Changes*.
Discard Undo Information will delete all undo information and apply all transformations to the font.

Preferences

The first option relates to CFF-based OpenType fonts:

— File Options

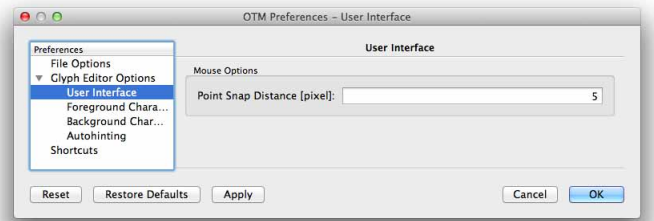
Subroutinize charstrings when writing 'CFF' table will compress the CFF table by finding common contour segments in all glyphs' outline descriptions, creating a dictionary of these, and referencing them.



The following set of options relates to the Glyph Editor:

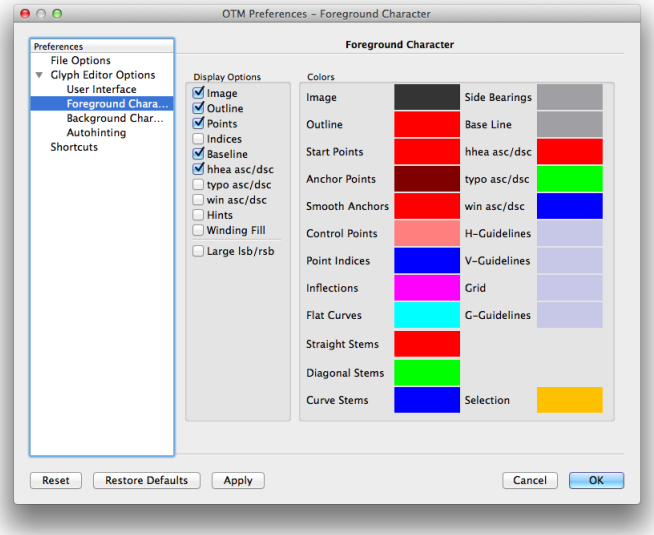
— User Interface

The single option **Point Snap Distance** defines how far away from an outline or point the mouse may be while still being able to select them.



— **Foreground Character**
— **Background Character**

There are two sets of options for **Foreground Character** and for **Background Character** the latter of which has a few options less:

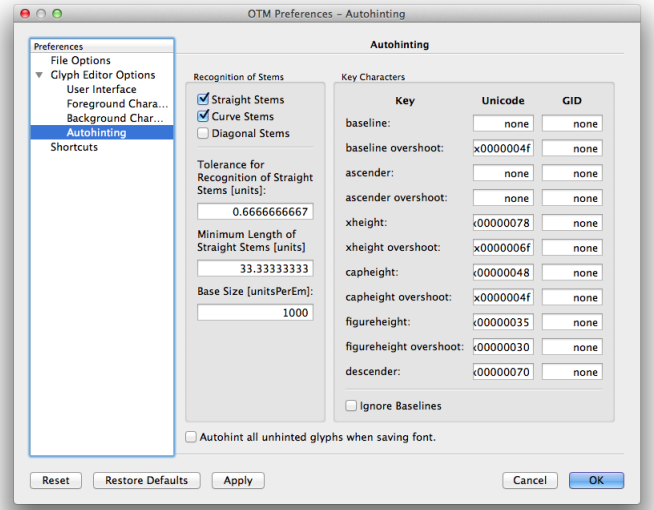


Checkboxes in the **Show Flags** area define which information the Glyph Editor will visualize by default:
Image shows filled shapes.
Outline adds a colored outline.
Points shows on-curve points.
Indices shows points' index numbers.
Box draws the bounding box around each glyph.
hhea asc/dsc are the hhea table's ascender/descender heights.
typo asc/dsc are the OS/2 table's sTypoAscender/sTypoDescender height.
win asc/dsc are the OS/2 table's usWinAscent/usWinDescent heights.
Hints shows a glyph's hints.
Winding Fill is the filling of outlines based on given contour directions.
Large lsb/rsb displays sidebearings by full-size vertical lines to the left and right of a glyph. Normally, and in best Ikarus tradition, sidebearings are indicated by small stubs sitting on the baseline.

In the **Colors** area you may adjust the colors of these pieces of information.

— Autohinting

These options determine how autohinting will work:



In the **Recognition of Stems** area you may choose whether or not to recognize **Straight Stems**, **Curve Stems** and **Diagonal Stems**.
Tolerance for Recognition of Straight Stems allows recognizing stems even if they are not exactly horizontal or vertical.
Minimum Length of Straight Stems defines how long straight segments need to be to be considered as being part of stems.
Base Size is the font's UPM size – the former two values relate to it.

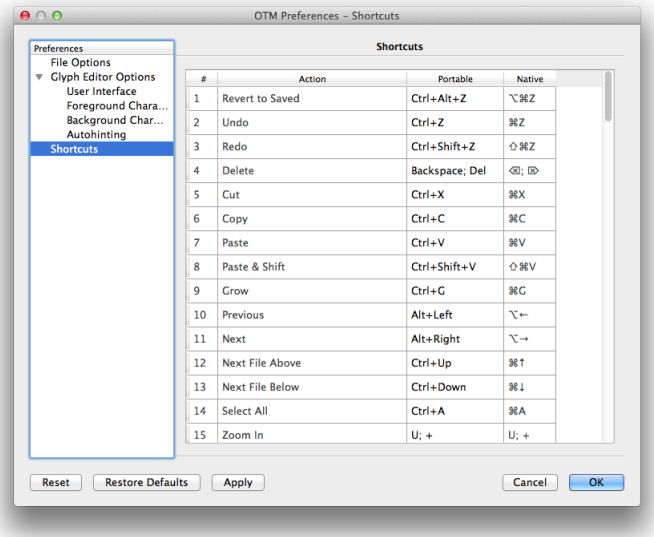
The **Key Characters** area offers an easy way to define alignment zones by referring to glyphs from which these values are to be derived. Glyphs are identified either by **Unicode** codepoint or by **GID**. The **baseline** value usually is zero and does not need to be extracted from any glyph.

By default, OTMaster requires that you autohint glyphs individually in the Glyph Editor. This is to make sure that existing hinting will not get lost. However, you may choose to **Autohint all unhinted glyphs when saving font**.

And finally there are the …

— Shortcuts

Here you may check, or redefine, which shortcuts are associated with which of OTMaster's actions:



Each action has a number, a name, a transcription of the shortcut (**Portable**) and a visualization of it (**Native**).

To adjust an action's shortcut, click into a cell in the **Portable** column and edit the cell's text. An action may be associated with more than one shortcut. In this case, a semicolon serves as a separator, as can be seen with action 15, 'Zoom In', which has two shortcuts enumerated as 'U; +'.

Function	Mac OSX	Windows
best fitting [values]	(⌘+B)	(CTRL+B)
Close	(⌘+W)	(CTRL+W)
Close All	(⌘+E)	(CTRL+E)
Copy	(⌘+C)	(CTRL+C)
Cut	(⌘+X)	(CTRL+X)
Delete	(⌫)	(⌫)
dec[imal values]	(⌘+D)	(CTRL+D)
Edit [GE]	(⌘+E)	(CTRL+E)
Fixup	(⌘+F)	(CTRL+F)
Grow	(⌘+G)	(CTRL+G)
hex[adecimal values]	(⌘+H)	(CTRL+H)
Nested Tables	(⌘+N)	(CTRL+N)
New [GE]	(⌘++)	(CTRL++)
Next [GE]	(⌘+N)	(CTRL+N)
Open	(⌘+O)	(CTRL+O)
Paste	(⌘+V)	(CTRL+V)
Preferences	(⌘+,)	(CTRL+,)
Previous [GE]	(⌘+P)	(CTRL+P)
Print …	(⌘+P)	(CTRL+P)
Quit	(⌘+Q)	(CTRL+Q)
Redo [GE]	(⌘+Y)	(CTRL+Y)
Reset [GE]	(⌘+R)	(CTRL+R)
Save	(⌘+S)	(CTRL+S)
Save All	(⌘+L)	(CTRL+L)
Save As …	(⌘+⇧+S)	(CTRL+⇧+S)
Scroll by Hand [GE]	(⌘+S)	(CTRL+S)
Select All	(⌘+A)	(CTRL+A)
Sort	(⌘+R)	(CTRL+R)
Text Dump	(⌘+T)	(CTRL+T)
Undo [GE]	(⌘+Z)	(CTRL+Z)
Zoom [GE]	(⌘+S)	(CTRL+S)

Functions marked with [GE] are available only in the ▶ *Glyph Editor*.

DTL *OTMaster Manual*
Edition 3.6/2013; supports DTL OTMaster version 3.6

Text,
Design Karsten Lücke
Typeset in DTL Argo, DTL Haarlemmer and DTL Haarlemmer Sans

Thanks to Microsoft for allowing the inclusion of the OpenType specs

Dutch Type Library
Zwaenenstede 49
5221 KC 's-Hertogenbosch
The Netherlands
phone +31 (0)73 614 95 36
fax +31 (0)73 613 98 23
e-mail info@dutchtypelibrary.com
website www.dtl.nl and www.fonttools.org

URW++ Design & Development GmbH
Poppenbütteler Bogen 36
22399 Hamburg
Germany
phone +49 (0)40 60 60 52 28
fax +49 (0)40 60 60 51 11
e-*mail* info@urwpp.de
website www.urwpp.de

FontMaster™ Utilities