



FontCreator 15

© 1997 - 2024 High-Logic B.V. All rights reserved.

FontCreator Manual

© 1997 - 2024 High-Logic B.V. All rights reserved.

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

FontCreator is a trademark of High-Logic.

Microsoft, Windows and OpenType are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Apple, the Apple Logo and Macintosh are registered trademarks and TrueType is a trademark of Apple Computer, Inc. registered in the United States and other countries.

Adobe and PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

All other trademarks and registered trademarks are the sole property of their respective owners.

The Unicode Character Database is provided as is by Unicode, Inc.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Table of Contents

Foreword	0
Part I Getting Started	9
1 Welcome to FontCreator 15	10
2 What's New in FontCreator 15	11
3 Getting Started with FontCreator	12
4 Technical Support	13
5 Registration	13
6 Credits	14
Part II Opening, Creating and Saving Fonts	15
1 Working with Font Projects	16
2 Open a Font	16
3 Create a Font Project	18
4 Save a Project	19
5 Export a Font	20
6 Close a Font	24
Part III Editing Fonts	25
1 Variable Fonts	26
2 Using the Font Overview	27
3 Glyph Names	29
4 Finding a Glyph	33
5 Used By	33
6 Insert Characters	34
7 Insert Glyphs	36
8 Copy and Paste Glyphs	38
9 Font Name	41
10 Font Family (typeface) Settings	42
11 Font Embedding	44
12 Monospaced versus Proportional	45
13 Unicode versus Symbol	45
14 Recommended Glyphs	46
15 Last Resort Fonts	50
16 Single Line Fonts	51
Part IV Editing Glyphs	55
1 Introduction	56

2	Glyph Metrics	57
3	Empty Glyphs	63
4	Simple Glyphs	63
	Introduction	63
	Contours	66
	Points	71
	Join and Split Contours	75
	Free Draw Tool	77
5	Composite Glyphs	78
	Introduction	78
	Add Glyph Member	79
	Glyph Member Properties	80
	Formula	81
	Complete Composites	82
	Auto Attach	84
6	Hybrid Glyphs	85
7	Color Glyphs	85
	Introduction	85
	Layered Color Glyph (COLR)	86
	SVG based Color Glyph	87
	Palettes and Colors	88
Part V OpenType Layout Features		91
1	Introduction	92
2	Types of substitution and positioning	92
3	OpenType Designer	95
	OpenType Designer	95
	Automatic OpenType Layout Features	99
	Class Manager	115
	Subtable Manager	116
	Designer Settings	117
	Autokern Settings	118
	Substitutions	119
	Single Adjustment	120
	Pair Adjustment	121
	Marks	124
	Cursive Attachment	126
	Anchor Manager	127
	Chained Context	127
	Feature Parameters	130
4	OpenType Proofing	133
5	Script Editor	135
	OpenType Layout Feature Code Editor	135
	Script Syntax OTLFD	138
	Basics	138
	Comments	139
	Script	139
	Language	140
	Class	141
	Feature	141

Feature Params	142
Lookup	143
Lookupflags	144
Subtable	144
Sub	145
Pos	146
Examples and Help	148

Part VI Font Tools 151

1 Tags	152
2 Guidelines	153
Options	153
Guideline	155
3 Metrics Options	156
4 Grid Options	159
5 Font Information	161
Calculated Fields	161
Unsupported Tables	163
6 Glyph Transformer	165
7 AutoMetrics	166
8 AutoKern	168
Setup	168
Kern	170
9 Import Images	172
Import Raster Image	172
Import Vector Image	174
10 Unicode Variation Sequences	180
11 Sorting Glyphs	182
12 OpenType Collection	183
13 External Tools	185
14 Printing	185
Print Font	185
Print Glyph	186
15 Font Validation	186
Setup	186
Results	189
16 Testing and Installing Fonts	190
Preview in FontCreator	190
MainType	193
International Keyboard	193
Test Your Font	194
Test Desktop Font	194
Test Font - Edit Text Samples	195
Test Web Font	196
Installing Fonts	196
Character Map	197

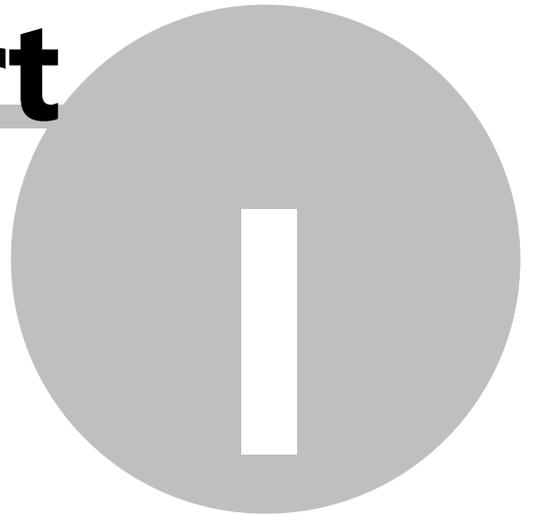
Part VII Toolbars and Panels 199

1	Overview	200
2	Font Properties	202
	Font	202
	Axes	211
	Masters	217
	Instances	233
	Naming Fields	240
3	Masters and Layers	244
4	Variations	246
5	Glyph Properties	246
	Glyph Properties	246
	Select Character	249
6	Transform	250
7	Validation	251
8	Preview	255
9	Background Image	256
10	Samples	257
11	Anchors	259
12	Palette	261
13	Color Glyph Members	262
Part VIII Customizing FontCreator		263
1	Options	264
	General	264
	View	264
	Preview	265
	Font	266
	Personalize	268
	Validation	269
	Exchange	270
	Advanced	272
2	Keyboard Shortcuts	273
3	FontCreator Data Files	279
Part IX About Fonts		281
1	TrueType	282
2	OpenType	282
3	Variable	283
4	Color Extensions	283
5	Web Open Font Format (WOFF)	285
6	FontCreator Project	285
7	Unified Font Object (UFO)	286
8	Designspace	287
9	Glyphs File Format	287

10 Font Copyright 288

Index 289

Part



1 Getting Started

1.1 Welcome to FontCreator 15

Introduction

FontCreator allows you to create and edit TrueType, OpenType and Web Fonts. It supports scalable color fonts and variable fonts, which allows you to make variable color fonts. The font editor lets you easily select and modify the entire character set. Features include the ability to convert images to vector-based outlines, thus enabling you to create fonts with your own signature, logo and handwriting.

The intuitive interface makes FontCreator the perfect tool for both new and experienced users. The advanced validation features make the design process easy and help you to avoid common mistakes. The OpenType Designer with interactive proofing allows you to easily add, edit, and debug OpenType layout features.

Key features

- Create and edit TrueType and OpenType fonts
- Full support for variable font technology
- Create and edit Web Open Font Format (WOFF and WOFF2) fonts with superior compression
- Create and edit scalable color fonts (both COLR and SVG with CPAL)
- Design fonts for engraving
- OpenType features are preserved upon opening a font
- Visual OpenType layout features
- Interactive OpenType proofing
- OpenType feature code can also be edited
- Redesign existing characters
- Add missing characters
- Convert vector and raster based images (e.g. a signature, logo or handwriting) to character outlines
- Edit or regenerate font names
- Fix character mappings
- Unicode variation sequences
- Generate, modify, and clean up kerning pairs
- Correct fonts that display incorrectly

- Add or correct composite glyphs
- Transform individual glyphs or an entire font (e.g. to make a bold version)
- Extract fonts from OpenType Font Collections
- Preview fonts before installing
- Install fonts in Windows

Make sure you have the latest version of FontCreator:

<http://www.high-logic.com/> 

1.2 What's New in FontCreator 15

New features in this version of FontCreator include:

- Elements -> supports outlines with both components and contours, so called hybrid outlines
- Redesigned Transform panel
- Support for the GlyphsApp .glyphs file format
- Improved shaping engine
- Improved detection and fixing of interpolation issues
- Improved auto attach
- Improved optimize contour feature
- Improved detection of invalid contour direction
- Improved join (union, intersection, exclusion, and also cut/knife) contours feature; including support for cubic curves
- Improved support for UFO and DesignSpaces
- Order axes
- Cancel changes in edit field (ESC key)
- Fixed remove overlap issue
- Increased stability and some speed gains
- Several other bug fixes and improvements

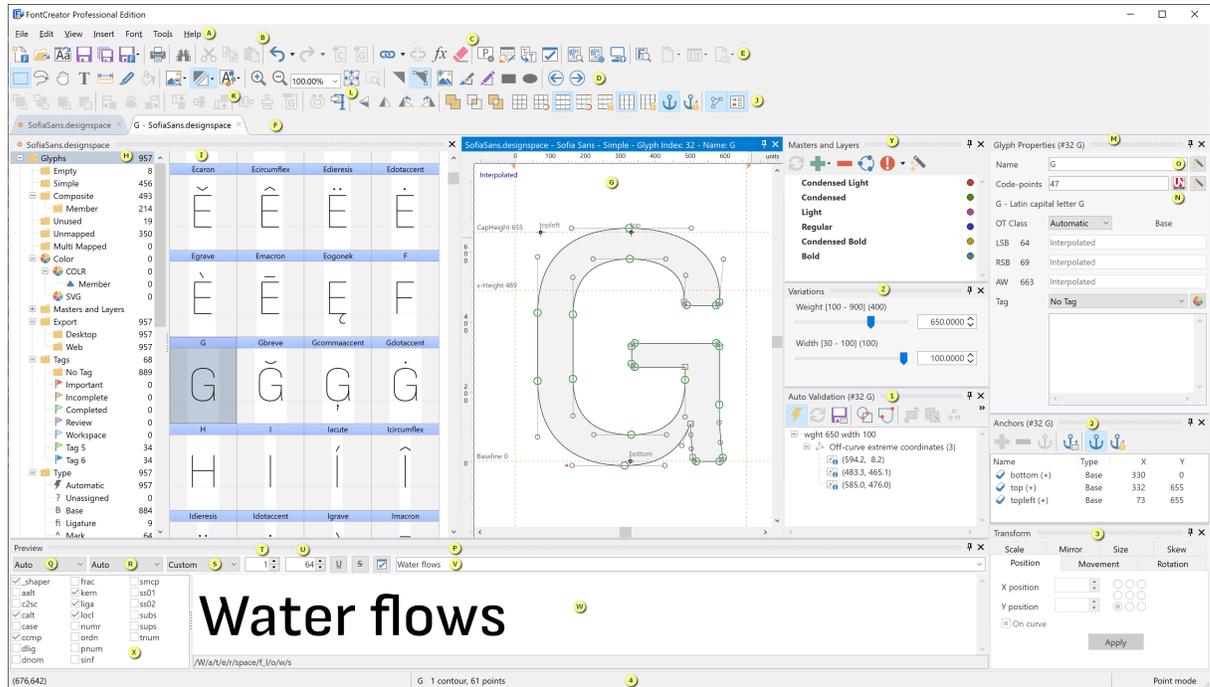
FontCreator is available in both 32-bit and 64-bit versions. The 64-bit version can allocate more memory and it is faster with time-consuming tasks like autokern, optical metrics, exporting fonts, and image to outline conversions.

A complete list of current and previous changes can be found online:

<https://www.high-logic.com/font-editor/fontcreator/changelog>

1.3 Getting Started with FontCreator

FontCreator comes with an advanced feature set, that makes it the tool of choice for professionals, and its intuitive interface is straightforward enough for users at any expertise level.



(A) Menu Bar, (B) Standard Toolbar, (C) Tools Toolbar, (D) Drawing Toolbar, (E) Overview Toolbar, (F) Font and Glyph Tabs, (G) Glyph Panel, (H) Font Categories Panel, (I) Font Overview, (J) Grid Toolbar, (K) Align or Distribute Toolbar, (L) Glyph Toolbar, (M) Glyph Properties Panel (N) Select Character, (O) Generate Name from code-point or Code-point from name, (P) Preview Panel, (Q) Script Drop List, (R) Language Drop List, (S) Feature Drop List, (T) Alternate Index, (U) Font Size, (V) Preview Text, (W) Preview Area, (X) OpenType Feature selection check-boxes, (Y) Masters and Layers Panel, (Z) Variations Panel, (1) Validation Panel, (2) Anchors Panel, (3) Transform Panel, and (4) Status Bar.

Within the font categories panel, there are several sub trees, among them are Color Glyphs, Scripts, and Unicode Character Blocks.

1.4 Technical Support

Online User Manual

The user manual is also available online:

<https://www.high-logic.com/fontcreator/manual15/> 

Forum

The forum is available to you for support and information about managing and designing fonts. This forum has become a place where all users of FontCreator can share their knowledge. Membership of the forum is free. There's a good chance other people have asked the same questions as you, so you may be able to find the answers you need. Feedback and suggestions are also welcome in the forum.

<https://forum.high-logic.com/> 

Frequently Asked Questions

View the most frequently asked questions (and answers) about FontCreator here:

<https://www.high-logic.com/font-editor/fontcreator/faq> 

1.5 Registration

Evaluation

During the 30-day trial period, FontCreator runs in **Professional Edition** mode, so all features are enabled. After using FontCreator for a 30-day trial period, you must register and pay for it, or remove it from your system. Registering makes your copy legal and helps support our efforts to develop innovative products to best serve your needs. Thank you for your support of FontCreator!

Why Register?

Registration allows you to continue using FontCreator 15 and entitles you to the following benefits:

- All future versions of FontCreator 15.x.
- Direct support from the developers.

Three Editions

FontCreator is available in three editions: a home edition with all basic features, a standard edition that will suit most font designers, and a professional edition to get the most out of your fonts. To view the differences between the FontCreator editions see the comparison chart:

<https://www.high-logic.com/font-editor/fontcreator/comparison-chart> ↗

To Register

The quickest and most convenient way to register is online via credit card. Other payment methods are also supported. For more details take a look at our order page:

<https://www.high-logic.com/buy-now> ↗

1.6 Credits

FontCreator contains several unmodified libraries that are available under the MPL.

- VirtualShellUtilities from <http://www.mustangpeak.net/>;
- Virtual Treeview from <http://soft-gems.net/>;
- ChromeTabs from <http://github.com/norgepaul/TChromeTabs>;
- SynEdit from <http://synedit.sourceforge.net>

FontCreator also uses an unmodified version of [ttfautohint](#) which is available under the FreeType License (FTL).

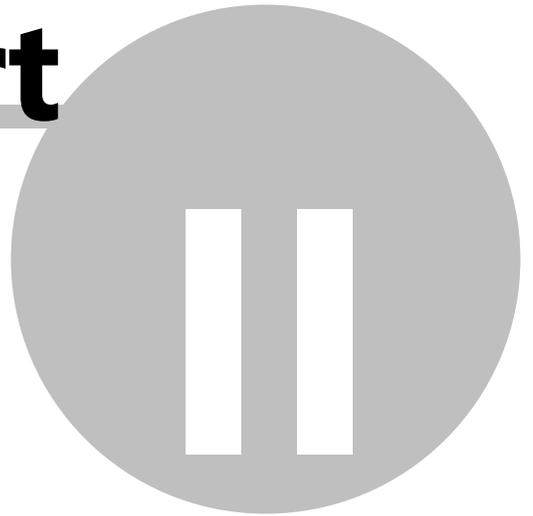
Some descriptions (of various fields in this document) are copied from the OpenType specification available online at:

<https://docs.microsoft.com/typography/opentype/spec/> ↗

Another important source is:

<http://www.unicode.org/> ↗

Part



2 Opening, Creating and Saving Fonts

2.1 Working with Font Projects

FontCreator supports several file formats to store fonts. Some formats are useful if you want to share your fonts with other people who use different font tools.

These font formats are supported by FontCreator:

- [FontCreator Project](#)
- [Unified Font Object \(UFO\)](#)
- [Designspace](#)
- [Glyphs File Format](#)

Our own FontCreator Project font file format is able to store all font data that FontCreator supports, so in general it is best to use that as your main source. From there, you can export fonts for use on desktop or web, as well as to interchangeable font project resources. This will make sure that regardless of the font format you export, all information about the font will remain available.

Opening existing project files

There are several ways to open existing project files:

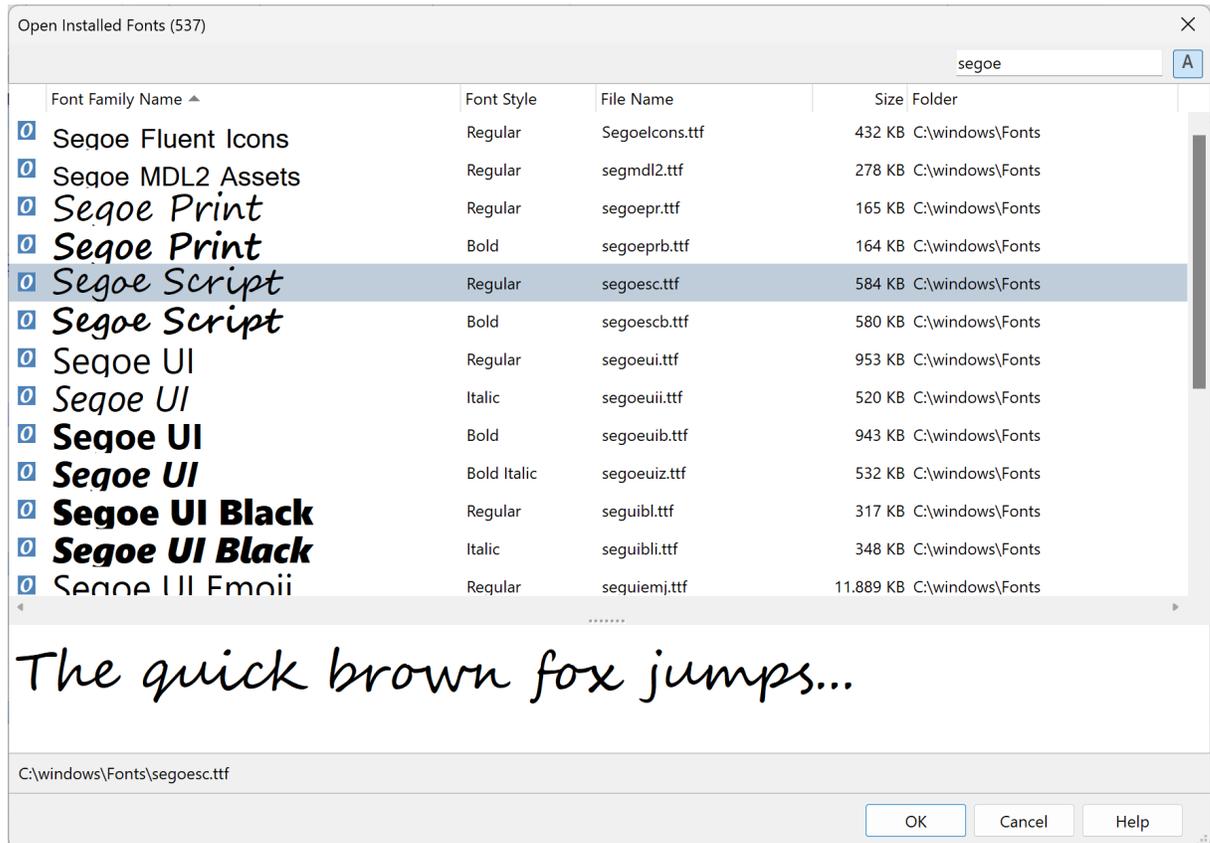
- Select Open... from the File menu
- Select a recent project from the Reopen submenu in the File menu
- Select a recent project in the Windows Taskbar Jumplist
- Double-click a project file in Windows Explorer
- Drag a project file from the Windows Explorer onto FontCreator

2.2 Open a Font

There are several ways to open a font file.

Open fonts already installed on your system

To open an installed font file select **Open** from the **File** menu and choose **Installed Font** option.



Open any font that is available

From the **File** menu, select **Open** and choose **Font File** option to open a font through Windows default open dialog box.

To open a UFO based font, select one of the files with a .plist extension. It will then process all required files from that folder.

Warning: If you want to open fonts that are located in Windows fonts folder you should use the Installed Fonts command (or drag and drop), because this folder is marked (by Windows) to be a special folder and therefore behaves differently (e.g. it shows font names instead of file names).

Windows Explorer File Associations

In the Windows Explorer window you can right-click a font or a font project and choose to open the file with FontCreator. Optionally this will let you always use FontCreator to open files with the specific file extension.

Drag and drop a font file

Another way to open a font is to drag a font file from Windows Explorer and drop it onto FontCreator.

Reopen a font file

To open a font that you've used recently, choose **Reopen** from the **File** menu to display the names of the last ten used fonts. Click on the font you want to use.

2.3 Create a Font Project

On the **File** menu, click **New...** to create a new font. The new font will contain more than 200 characters, and covers over 100 languages. You can further extend the font with [more characters](#), or remove characters to suit your needs. Several [transform scripts](#) will help you add numerous characters to support more languages.

New Project

Font family name
My New Font

Font style
 Regular Italic Bold Bold Italic

Preferred outline format
 TrueType (quadratic) CFF / PostScript (cubic)

Predefined outlines
To speed up creating your new font it is recommended to include outlines for several common characters.
 Include outlines Don't include outlines

OK Cancel Help

Font family name

In Windows, the Font family name is displayed in the font menu. The **Font family name** will appear as the font name when you select a font in a word-processing program, etc.

Font style

The font will be identified through its **Font family name** and the **Font style**. To create a full font family, you will need to create four fonts, each with a different **Font style**, but with the same **Font family name**.

Preferred outline format

Here you can choose if you want to design outlines with quadratic or cubic based contours. Several outline based features like paste or import outlines will depend on this setting. Some other initial font settings like units per em will also rely on this option. You can always change the outline format through the Export Settings dialog.

Predefined outlines

Selecting **Include outlines** will add outlines for several common characters. This will speed up creating your new font. You can replace or modify these glyphs later. The outlines can be used royalty free in your own fonts.

Note: On the **Personalize** tab, accessed by **Tools -> Options**, there are default naming values that will be added to the new font.

See also:

[Default Naming Values](#)

[Insert Characters](#)

2.4 Save a Project

To save an active project select **Save Project** in the **File** menu. If you want to save the active project with a different name, or in a different location, select **Save Project As** in the **File** menu, choose a name and location and click **Save**.

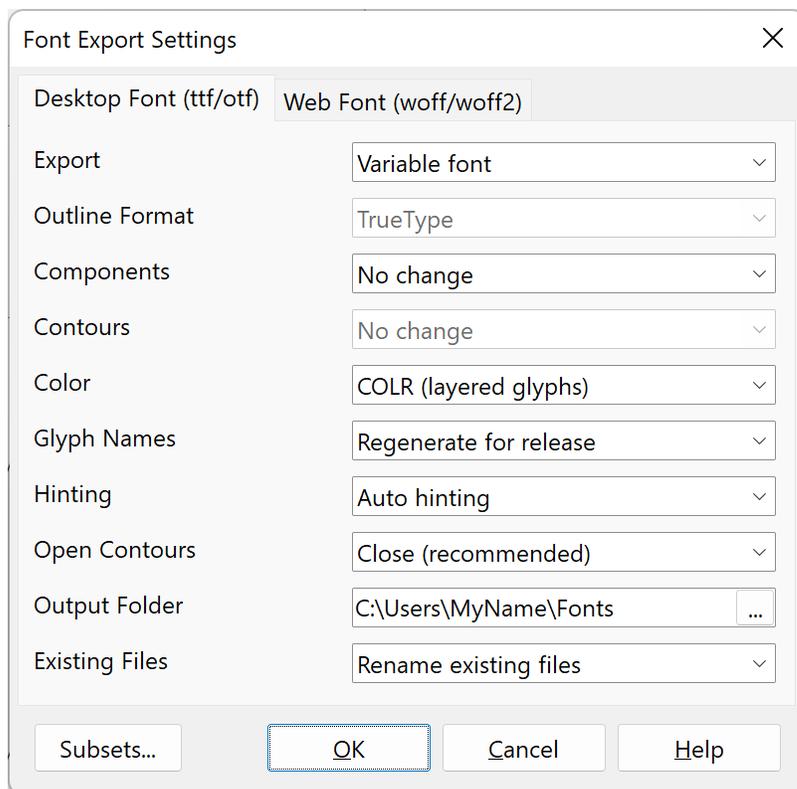
2.5 Export a Font

To export an active project to a Desktop (ttf.otf) or Web (woff/woff2) font select one of the export formats in the **Export Font** submenu in the **File** menu. All available OpenType layout features will be embedded in the font.

Note: Due to technical limitations within the OpenType font specification, the number of glyphs that can be exported to a font is limited to 65535.

Note: only glyphs that are marked as included for a specific export will be exported. To exclude glyphs, select them within the Font panel, right-click and select Include in Exports -> None.

The first time you export your font you will be asked to choose an output location. If you want to export the active project in a different location, select Export Font As in the File menu.



When you open an existing font, the export settings will be set in a way that they match the original font properties as much as possible.

Export

This option is available for variable fonts. It allows you to export the variable font, font masters, font instances, or the current location.

Outline Format

TrueType - The font is exported with TrueType based outlines (quadratic Bézier curves)

CFF (PostScript) - The font is exported with Compact Font Format (CFF) based outlines (cubic Bézier curves)

Components

This option allows you to decompose composite glyphs into simple glyphs. It is common practice to make use of composite glyphs, but they can cause rendering issues if one or more [glyph members](#) make use of scaled, rotated, or transformed properties. Set this option to **decompose scaled** to ensure compatibility with all rendering engines.

Contours

While it is common practice to have overlapping contours during development of outlines, it is something that is usually avoided with non-variable fonts.

Color

No color - The font is exported without color information

COLR (layered glyphs) - The font is exported with layered color information (both COLR and CPAL)

SVG without color palettes - The font is exported with SVG based color information (SVG)

SVG with color palettes - The font is exported with SVG based color information (SVG and CPAL)

Both COLR and SVG - The font is exported with both color extensions (COLR, CPAL, and SVG)

Here you can see where the [color extensions](#) are supported.

Note: if CPAL is included, all [color palettes](#) are stored within the font.

Glyph Names

While developing fonts, you can define user friendly glyph names, which help you quickly identify the glyphs and it allows you to share OpenType layout feature code.

When your font is ready for release, FontCreator can automatically rename glyph names to recommended names. Those names might be important for accurate interpretation and reconstruction of underlying Unicode text encoding with some PDFs. Since in most cases glyph names are no longer important, you can decide to leave them out to reduce the file size. We even recommend omitting them in WOFF fonts to reduce the file size.

Note: Glyph names are still required (and thus always included) in CFF based fonts, but even that will likely change in the future.

Hinting

Hinting information will improve readability on screen for smaller font sizes.

No Hinting - No hinting is added to the font.

Autohinting - Hinting information is automatically generated.

Keep Original - Stores hinting information that was originally available in the font.

Note: Hinting is not available for CFF outlines

Open Contours

Note: As this option can be confusing to novice users, this option is hidden by default. It will become visible when **Enable open contours (design time only)** is checked on the [Font tab](#) in the Options dialog.

Exported fonts can't contain open contours, as the specification only allows for closed contours. However it is possible to design your glyph outlines with open contours. This might be useful in certain circumstances. Especially for [designing single line fonts](#) for laser cutting and CNC milling, as then FontCreator will show contours as single stroke outlines.

Close - Open contours will be closed within the exported font. The Toggle Open Contours feature within the Glyph panel is hidden. This is the default as it will generate fonts that conform to the OpenType specification.

Exclude - Open contours will not end up in the exported font.

Single Stroke - Only works with TrueType based outlines in combination with software that supports "single line / single stroke / open loop / stick fonts", e.g. Rhino, SolidWorks.

Double Stroke - Only works with TrueType based outlines. The path of all open contours will get additional points, so the path is also reversed in the exported font. Double stroke fonts work with most engraving machines.

Note: In FontCreator both Single Stroke and Double Stroke will show glyph outlines very different as contours will not be filled. Fonts generated this way are mainly meant for engraving. Single Stroke fonts will look awkward in standard software, and Double Stroke fonts will most likely show thin lines in standard software.

Output Folder

Location where the font file(s) will be exported. If this field is left blank, FontCreator will show a save dialog the first time you export the font.

Warning: It is recommended that you do not export your font directly into the Windows fonts folder.

Warning: Files locked by Windows can't be saved. This happens when the Operating System keeps the font in memory. You could try to uninstall (delete) the font through the Windows fonts folder.

Existing Files

Use this option to avoid conflicts with existing files in a folder.

Subsets

This option is available for variable fonts. FontCreator allows to export multiple subsets out of a variable font. Each subset can have its own default location and you can even extrapolate by extending minimum and/or maximum values of an axis. Add each axis that you want to include in the subset. Not mentioning an axis is equivalent to slicing the space at the default value of that axis. Values are in user coordinates.

Note about the file extension:

Desktop fonts may have the extension .otf or .ttf depending on the type of outlines in the font (see Outline Format below) and the desired backwards compatibility. A font file with TrueType outlines should have either .otf or .ttf extension, depending on the desire for backward compatibility on older systems or with previous versions of the font. A font file with CFF (PostScript) outlines should have file name extension .otf.

In all cases, software must determine the kind of outlines present in a font not from the file name extension but from the contents of the file.

Legacy TrueType Fonts

Sometimes it's preferred to export a font in the old TrueType format for older software. You probably need to include legacy font data in the [Options dialog](#) and set the export settings to the following settings:

Outline Format: TrueType

Glyph Names: Regenerate for release

Hinting: No Hinting

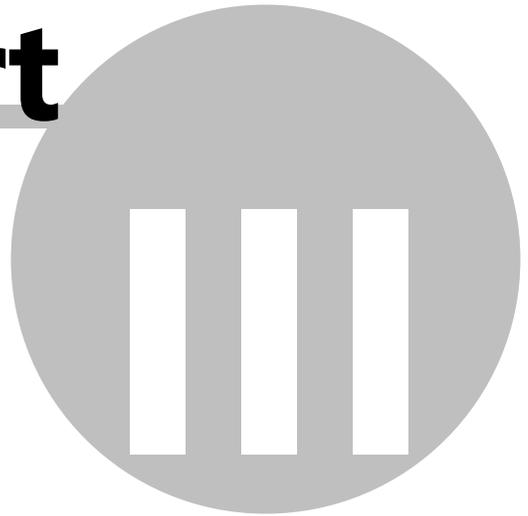
Legacy kern table

Cutting edge fonts store kerning data as pair adjustment lookups in OpenType layout features, while two decades ago fonts used to store basic kerning data in a kern table. FontCreator no longer supports this legacy kern table.

2.6 Close a Font

To close the current font or project select **Close** in the **File** menu. To close all active fonts and/or projects at once select **Close All** in the **File** menu.

Part



3 Editing Fonts

3.1 Variable Fonts

Note: Feel free to skip this section, if you only intend to make and edit non-variable fonts.

A variable font allows you to use a continuous range of style variations by means of interpolation. Even if you don't want to make an actual variable font, this is also the most convenient way to make several styles within a single font family, as you can export all pre-defined styles (named instances) with a single click.

To make a variable font, you need to include at least one [axis](#), the default master that holds the default glyph outlines, and one other master positioned at a different location on the axis.

Most of the required properties are located on the [Font Properties panel](#) (CTRL+F2).

It is recommended to first take some time to decide what axes and masters you need for a specific variable font, and where you intend to have the default master. Other fonts might be an inspiration, but there is also a lot of information about design spaces found online.

Editing Across Layers

The Masters and Layers panel has a "Edit Across Layers" toolbar button, which when enable will perform several operations, like delete points, for all glyph layers at once. For most of such actions, it is required that layers are compatible. If you only want to edit the active layer, then make sure the "Edit Across Layers" option is not enabled.

It is common practise to have overlapping contours in a variable font, but there are several pitfalls.

Artifacts where segments overlap

To avoid artifacts while rendering such outlines, always ensure contours do not have partly equal paths.

Kinks

A kink can appear in interpolation. They occur if all these conditions are met between two masters:

- Three (or more) points have to stay in one line
- The angle is different
- The ratio of both line lengths are different

The kink is largest in the middle of the masters.

Overlaps within a contour

Overlapping contours that reside within another contour can cause rendering issues, as the rasterizer assumes the parts that overlap should not be filled. In such case consider removing overlaps.

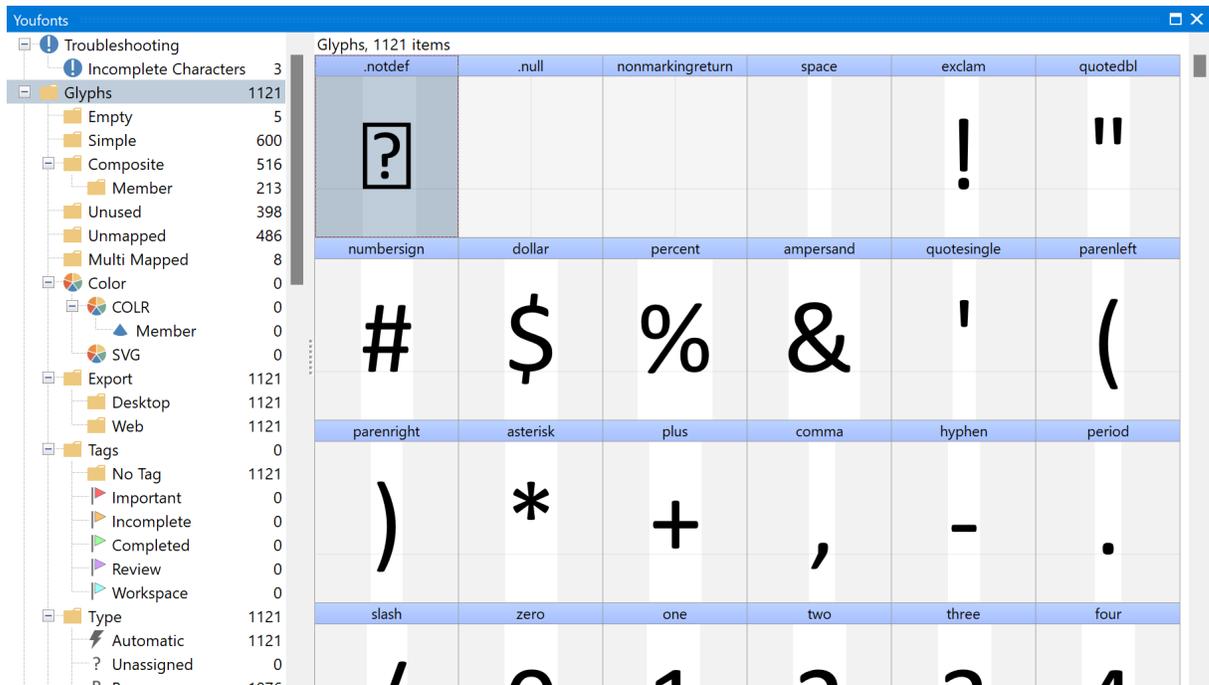
Point Reduction

To reduce the number of points that end up in a font, it sounds tempting to remove all overlaps that do not influence interpolation before exporting the font. This is a manual process, and usually is not worth the trouble.

For more information about how to make variable fonts, do take a look at the online [FontCreator Tutorials](#).

3.2 Using the Font Overview

In the **Font** panel there is a list of categories and a grid where all cells have a caption and a part that shows the glyph. A cell that contains an empty character will show a sample character with light grey outlines. Sample characters don't really exist in the font, but help you identify what characters are missing.



The categories panel is a convenient way to quickly show a subset of characters or glyphs and allows you to display them in several groupings and orderings. To change the current grouping and ordering use the **Overview Toolbar** on the main toolbar.

The Glyphs category contains several subcategories. One of them is the Unused category, which is only shown when the font contains one or more glyphs that are inaccessible by normal means. It is recommended to exclude such glyphs from your fonts as they are of no use. This setting is located on the [Options -> Font tab](#).

The Scripts category reveals all scripts that the font covers. Numerous characters are used among several scripts (digits, punctuation, etc), hence they are excluded.

Each cell has a caption that is used to display the glyph name, Unicode name, the code-points or the glyph index. To select the kind of caption, use the **Overview Toolbar** or right-click in the **Font** panel and select a specific caption. Setting the caption type to **Automatic** will automatically change the captions to the value of the current selected grouping method. This means that when you group by Advance Width, the cell captions display the **Advance Width** value.

You can also switch between decimal and hexadecimal values for the code-points. This setting is located on the [Options ->View tab](#).

3.3 Glyph Names

Glyph names are multipurpose, as they are used with FontCreator to identify glyphs and help with generating OpenType features. Glyph names can also be stored inside the generated font, but that is optional.

Years ago glyph names were an important part of a font, but nowadays most fonts are shipped without glyph names, as that information is no longer used, thus only takes up space.

Glyph names within FontCreator are therefore mostly used as a way to show friendly names of your glyphs and to [generate OpenType features](#). The names can also be used in [glyph metrics expressions](#), and are also used when you import or export OpenType layout feature scripts, so it is important to give your glyphs proper names; mainly for development purposes.

Friendly Glyph Name Convention

FontCreator uses friendly names for all Unicode characters. They allow you to quickly identify them by script and optionally by the intended OpenType feature.

Unicode Characters

The name starts with the actual friendly description of the character (or ligature) and optionally an abbreviation for the script (which starts with a hyphen) and one or more suffixes (all start with a dot). The script is dropped if it is Latin. Here are some more examples:

Friendly Name	Code-point	Unicode description
Ccedilla	\$00C7	LATIN CAPITAL LETTER C WITH CEDILLA
five	\$0035	DIGIT FIVE
A-cyrl	\$0410	CYRILLIC CAPITAL LETTER A
fi	\$FB01	LATIN SMALL LIGATURE FI
zacute	\$017A	LATIN SMALL LETTER Z WITH ACUTE

Ligatures without Unicode Code-Point

Ligature glyphs have a name that is actually a concatenation of the glyphs that make up the ligature with an underscore between the individual glyph member names.

Friendly Name

f_f_j

z_gravecomb

sheen_yehhamzaabove-arab.fina

Alternate Forms

If you want to design an alternate form, it is highly recommended to add a glyph with the same name, but with a suffix. If you want to use it in a specific OpenType feature. Examples:

Friendly Name

Ccedilla.ss01

five.sup

A-cyrl.pcap

zero.zero

If a glyph name is used in an OpenType script, it could cause issues with the parser if it contains specific characters. One of them is the hyphen which is both used to provide a script to the glyph name, but also to define a range of glyphs. In such case the glyph name comes between double quotation marks.

Tip: If you wish to include glyph names in your generated fonts, it is best to select "Regenerate for release" in the Export Settings. This way you can provide friendly glyph names to be used for production only.

Warning: Even though FontCreator allows you to have two or more glyphs with the same name, it is bad practice, confusing, and exporting OpenType layout feature scripts can cause ambiguity.

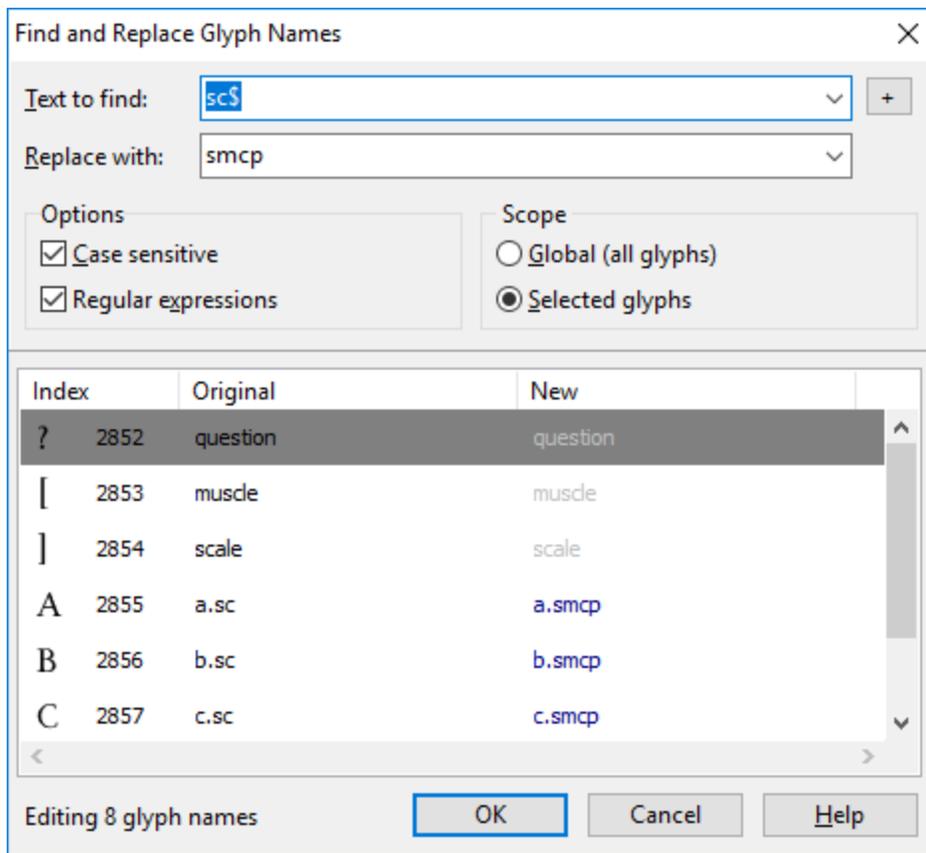
Generate Glyph Names (available through the main menu -> Tools -> Glyph Names -> Generate) will generate glyph names for all glyphs with a known code-point. It will also try to set names for glyphs that are used in the specified OpenType layout features.

Note: You can override those names by adding entries in the `glyphnamesnew.dat` file in the user data folder.

See [FontCreator data files](#) for more information.

Use the Edit Glyph Names dialog to quickly replace glyph names. You can type, copy, and paste from the left side text area.

Use the Find and Replace Glyph Names dialog to replace parts of glyph names. Optionally you can use regular expressions. That way you can easily prefix your glyph names with some text, or append text to the glyph names. Use the [+] button in the upper right corner to access common regular expressions.



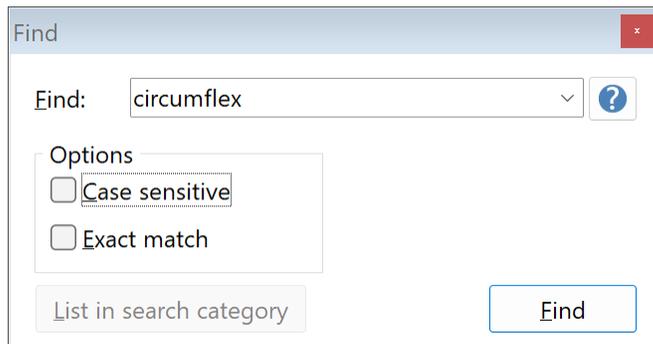
As you can see in the above screenshot, regular expressions allow you to search for text that ends with "sc" (the dollar sign \$ marks the end of a line) and replaces it with "smcp". Without regular expressions, the glyphs with names muscle and scale would also be renamed as musmcp and smcpale.

To append ".case" to the end of all glyph names, use regular expressions with Text to find set to "\$" and Replace with set to ".case".

To easily identify changes; the text in the New column is grayed if the text hasn't changed. It will be shown in blue if the glyph name will be changed.

3.4 Finding a Glyph

You can search for glyphs and or characters by their glyph name and mappings. Select **Find** on the **Edit** menu to open the **Find** window or use the keyboard shortcut Ctrl+F.



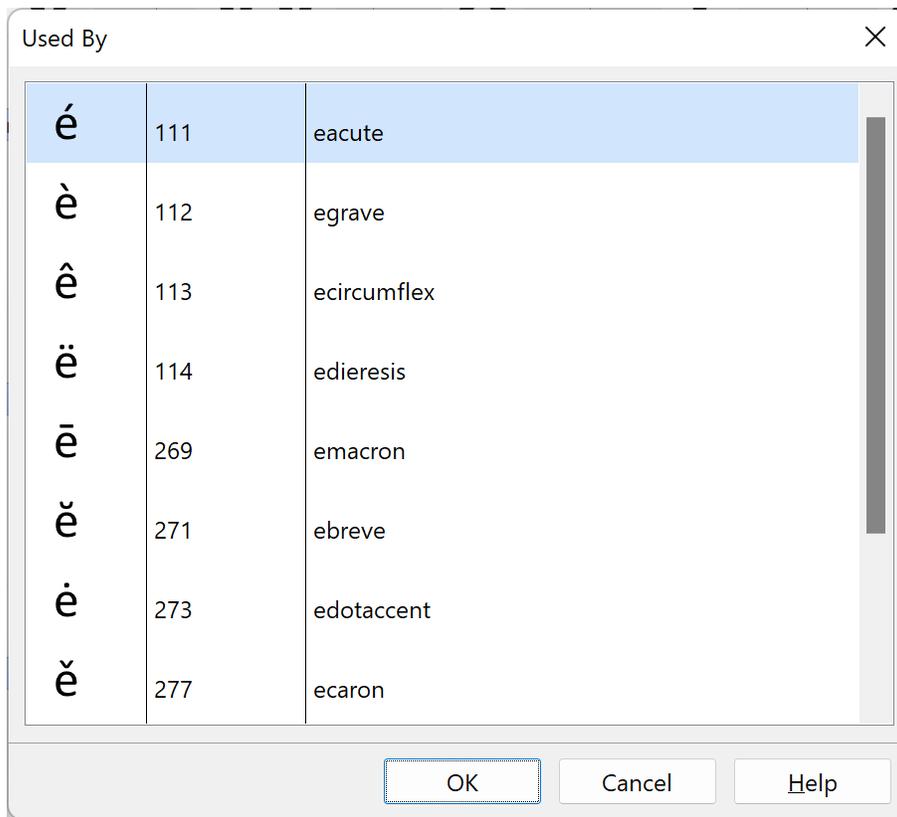
The input field accepts several kinds of keywords:

- Single character : Entering a single character will find the entered character if it exists in the font. (Example: "a") Note that this type of search is case-sensitive regardless of the case-sensitive checkbox!
- Part of glyph name
- Start with: "A*" will return Agrave, Aacute, etc.
- Ends with: "*grave" will return Agrave, Ugrave, etc.
- "#353" will show that exact glyph index
- Unicode ranges \$0032-\$0046, \$0062, 65, 214, etc. (can be entered in decimal or hexadecimal)

List in search category will add all matching glyphs in a new search category in the categories panel.

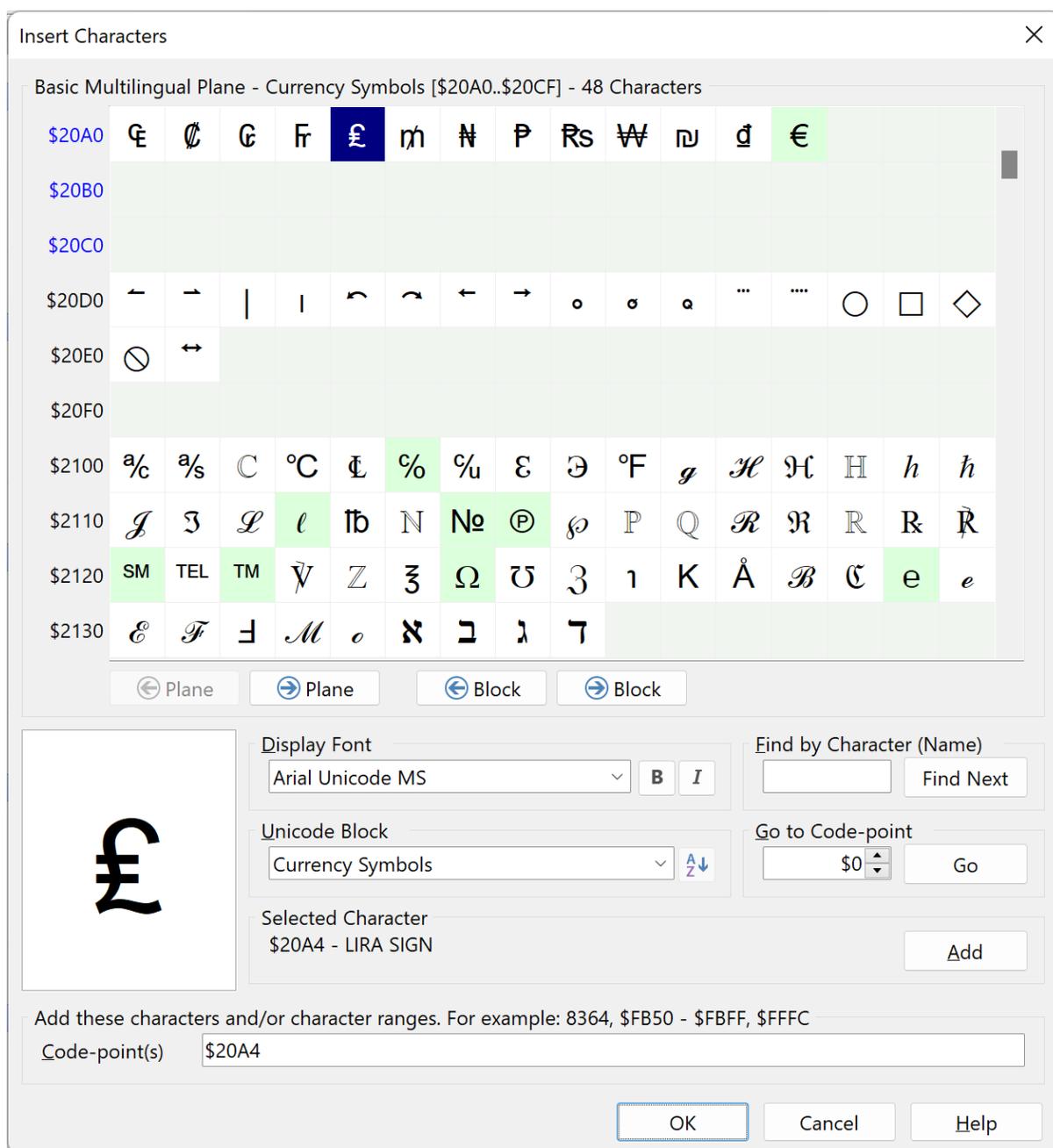
3.5 Used By

The Used By window (available by right-clicking a glyph in the **Font** panel or **Glyph** panel and selecting the **Used By** menu item) is used to display an overview of all glyphs that use the selected glyph. To jump directly to one of the displayed glyphs double-click it in the window.



3.6 Insert Characters

Select **Characters** in the **Insert** menu to add glyphs with their character mappings and glyph names to the font. This option is available when the **Font** panel is active.



Select a Unicode block from the combo box. Deselect the “AZ” button to sort the blocks in numerical code-point order instead of alphabetical order. If necessary, choose another installed font which includes the characters to be added. Characters can be added anyway, but it helps if the glyphs can be previewed.

Use the next and previous block or next and previous plane buttons to scroll through the font. The code-point of the character to add can be found by entering the

decimal value in the “Go to code-point” field, or by entering the hexadecimal value preceded by a dollar sign or 0x. For example, enter 8364, \$20AC, or 0x20AC to find the Euro Sign (€).

Either a character or (part of) the Unicode name of the character can be used in the field, “Find by Character (Name)”. For example, type “€” or “euro” to find the EURO SIGN. When enter a character it will find an exact match, but with part of the Unicode name you can find another match if you press the Find Next button again.

Click on a character to select it and preview it in the glyph preview at bottom left. Double-click the character, or click the Add button, to add its code-point to the list of selected characters at the bottom of the dialog. Keep adding individual characters by double-clicking, or hold down the Shift key and double-click to add a range of characters. The code-points will be displayed in Hexadecimal or Decimal notation depending on the setting in **View -> Display**. You can also type code-points into the characters to add field, separated by commas (or hyphens to add a range of characters), or cut and paste a predefined list of characters from a text file. For example, pasting 256-383 then clicking OK would add the entire Latin Extended-A character set.

Click OK to dismiss the dialog and add the characters to the current font. If the glyphs exist in the overview sample font, and if [Show samples in empty glyphs](#) is on, grey outlines of the new characters will be displayed in the font panel.

Characters or entire character sets can also be added using [Transform Scripts](#). See the topic: Glyph Transformer.

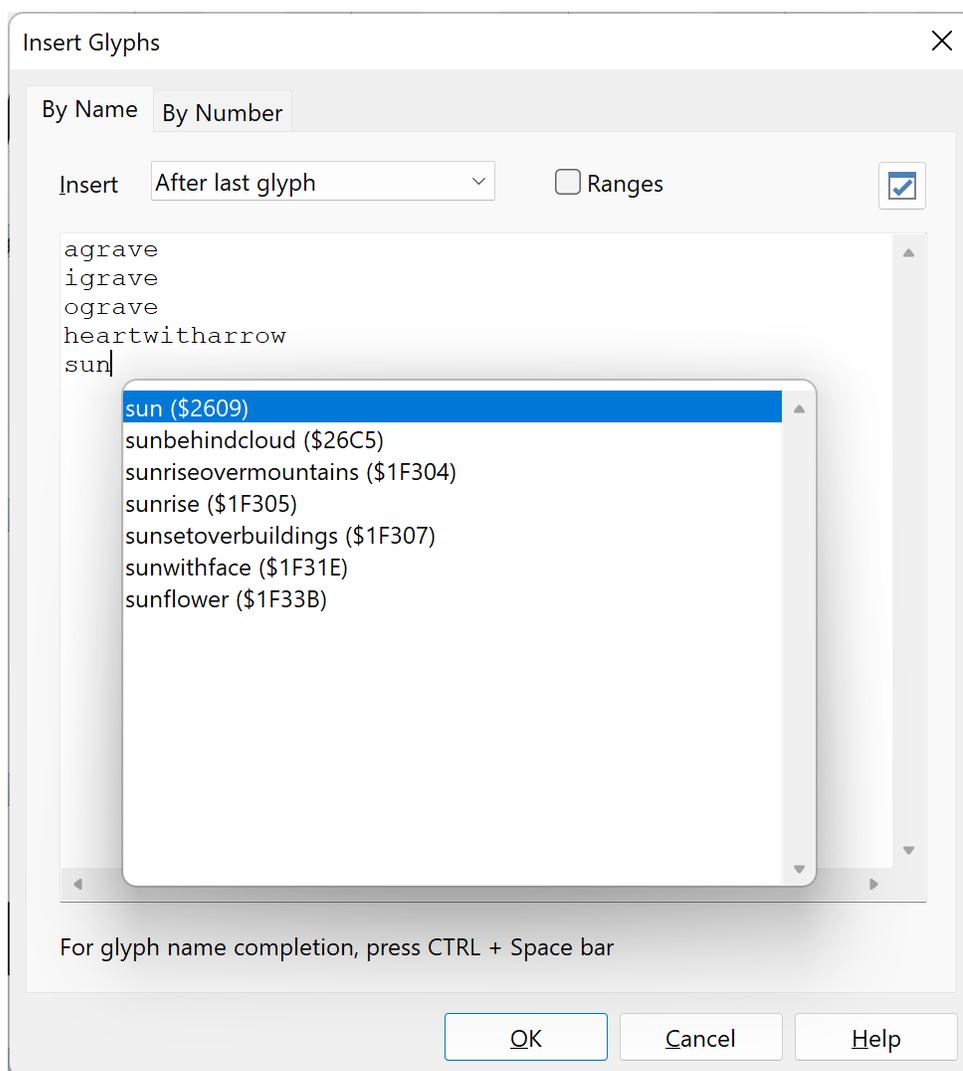
Note: The number of glyphs that may be included in one font is limited to 65535.

See also:

[Glyph Transformer](#)

3.7 Insert Glyphs

Select **Glyphs** in the **Insert** menu to add glyphs to the font. This option is available when the **Font** panel is active.



Press CTRL + Space bar to show the glyph name completion helper.

FontCreator allows you to add ranges of glyph names. If a glyph name matches a Unicode character, that character will be inserted. These ranges are supported:

- a-z
- uni2000-uni20FF
- uni0391.pcap-uni03A9.pcap
- u2000-u20FF
- u102000-u1020FF

- \$2000-\$20FF
- 0x2000-0x20FF

These are not allowed:

- Cdoublestruck-Hdoublestruck
- elevencircled-twentycircled

Generate Composites with a Formula

For each glyph name you can also provide a [formula](#) for generating the glyph outlines as a composite. Just provide the glyph name, followed by an equal sign and the glyph names separated by a plus sign. Don't use spaces as that would be treated as a new glyph you want to insert. Here is an example:

f_i_j=f+i+j

It will add a glyph named f_i_j and will make a composite that is made out of these glyphs: f, i, and j.

Note: in some cases it might be unclear if a hyphen is part of a glyph name or meant as a range indicator. Then either disable the Ranges option, or provide the name and formula within double quotation marks, as shown here:

```
"beh_yehhamzaabove-arab.fina=alefmaksura_alefmaksura-arab.fina+symboldotbelowmod-arab+arabichamzacombo"
```

See also:

[Formula](#)

3.8 Copy and Paste Glyphs

You can copy and paste many different parts of glyphs. You can select multiple glyphs from within the Font panel and the copy them to the clipboard. This way you can paste them into another font or within the same font at a different selection of glyphs.

You can also copy (part) of the glyph outline within a glyph panel. Then you can paste it within the same or another glyph panel, but you can also paste this outline data in vector-based software like Adobe Illustrator.

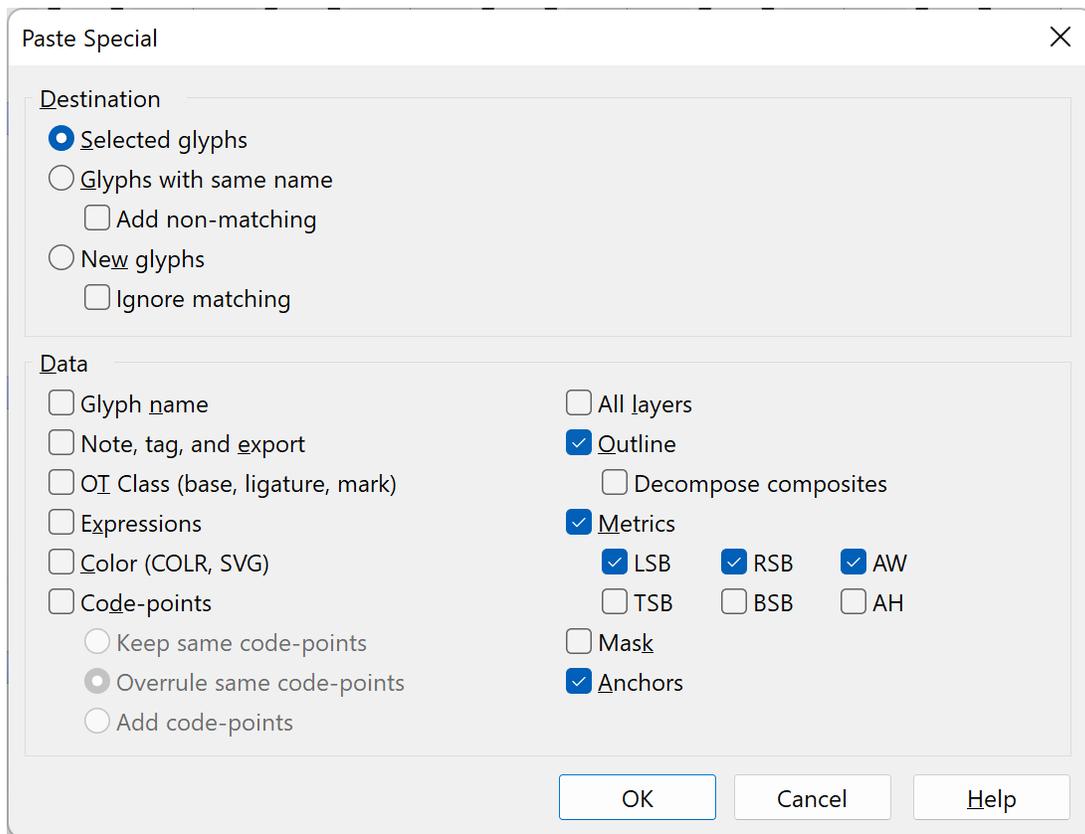
See: [Import Vector Image](#).

The **Paste Special** command is used to specify what parts of a group of glyphs (already copied to the clipboard) should be pasted.

You can select any number of glyphs in the **Font** panel by clicking on them while holding down the **Ctrl** key. You can perform several operations on the selected glyphs.

Destination allows you to paste the glyphs into:

- The current glyph selection; select the same number of glyphs you copied prior to the paste action, or select just one glyph to copy all starting at the selected glyph. If you copied a single glyph, it will be pasted into all selected glyphs.
- Glyphs with the same name (optionally add if not already available)
- Append as new glyphs (optionally ignore if already available).



You can specify what to paste through the options in the **Data** section.

Glyph Name will paste all glyph names

Metrics are the side-bearings and Advance Width of each glyph; The side-bearings option includes left, right, top, and bottom side-bearings.

Code-points will paste the mappings. Keep same code-points adds new mappings to the glyphs. Mappings that already exist will be reassigned to the pasted glyph(s) when the **Override same code-points** option is selected. You can also choose to add mappings. **Add code-points** adds all mappings. When mappings are already available, they will be reassigned to the pasted glyph(s).

Anchors will include all available anchor points which can be used by OpenType layout features.

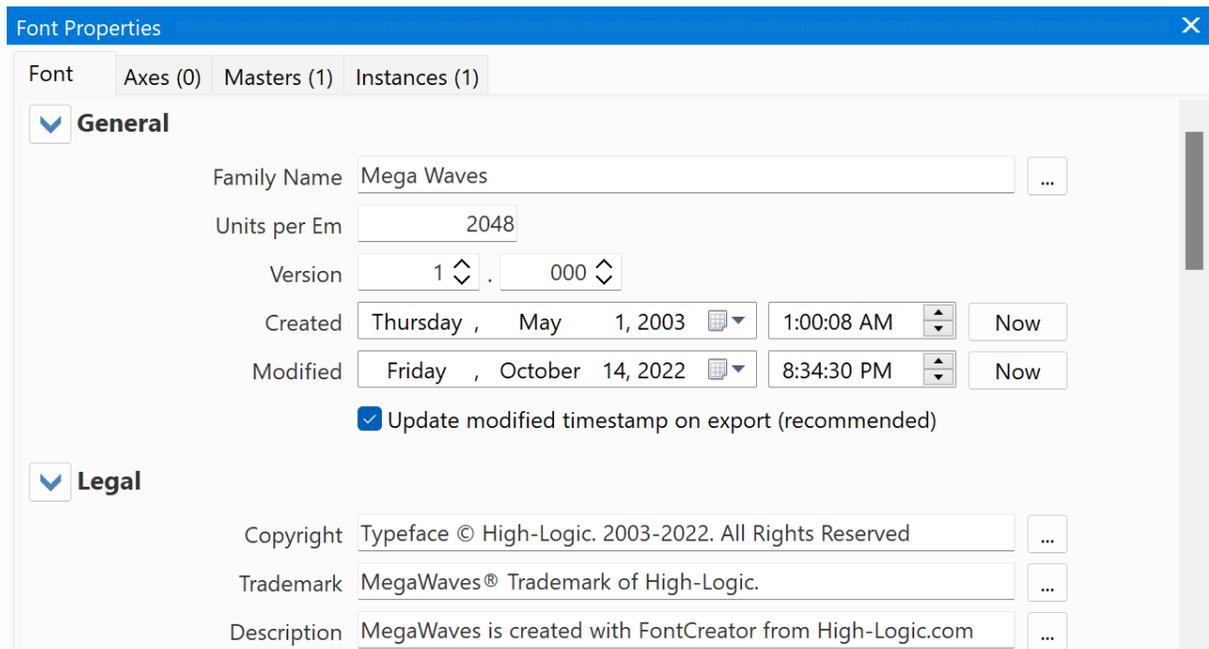
Note: copy and paste composite glyphs and/or color outlines between fonts relies on glyph names.

3.9 Font Name

Be careful not to confuse the font name with the file name. Windows uses the file name to install a font, while the font name is used to identify the font.

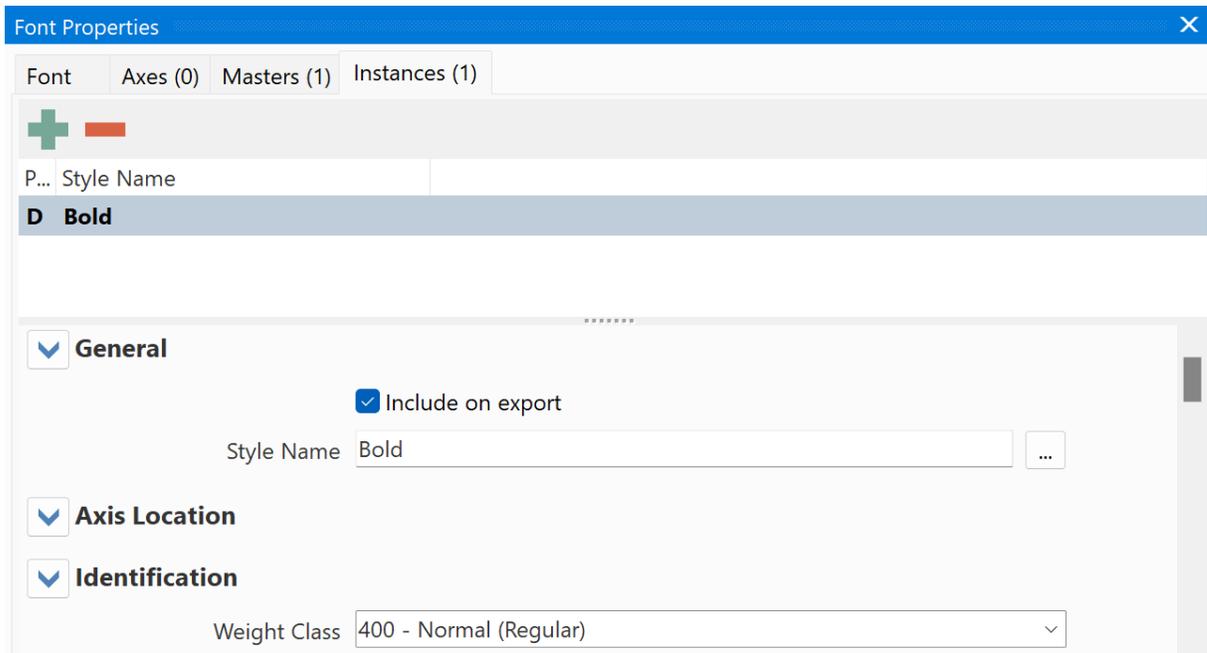
To change the font name select the Font tab from the Font Properties panel and edit the **Family Name**.

For non-latin fonts, like Chinese, Japanese, Korean, Hebrew, Arabic, etc, you can add localised names, accessed by pressing the [...] button.



In case you also want to change the style name, select the Instances tab from the Font Properties panel, make sure the default instance is selected (and include on export is checked), and edit the **Style Name**.

For non-latin fonts, like Chinese, Japanese, Korean, Hebrew, Arabic, etc, you can add localised names, accessed by pressing the [...] button.



3.10 Font Family (typeface) Settings

Note: This topic assumes you have a non-variable font, thus a font without axes, and you don't want to turn the font into a variable font.

If you want to make a bold font out of a regular font, you also need to adjust all your glyph outlines. This can be easily accomplished through the [transform features](#), using the Medium to Bold script. There are several other scripts that can make a font thin, italic, etc.

If the font already contains bold outlines, but is recognized as a regular font, then you need to change a few settings on the **Font Properties** panel:

Instances Tab:

- Style Name: Regular -> Bold
- Weight Class: 400 - Normal (Regular) -> 700 - Bold
- PANOSE: Optionally change Weight from 5 (Book) to 8 (Bold)

How to make fonts work like a family?

Windows comes with several typefaces (font families) that are made up with more than 4 fonts, so such family comes with more than just Regular, Italic, Bold, and Bold

Italic. To name a few; Calibri contains 6 fonts, Segoe UI comes with 12 fonts and Sitka is king with 24 fonts. The trick to make this work is fairly straightforward; however you need to be consistent. All fonts within the family must have the same Typographic Family Name. However the four common fonts (Regular, Italic, Bold, and Bold Italic) leave the Typographic Family Name and Typographic Subfamily Name empty, and instead will use the Style Map Family Name and Style Map Style Name fields. The other fonts will be grouped together based on their Width and Weight, and will get their own Family Name based on that.

Older software typically supports the combination of Style Map Family Name and Style Map Style Name, while modern software uses Typographic Family Name and Typographic Subfamily Name. If the Typographic Family Name and Typographic Subfamily Name fields are empty, the values from Style Map Family Name and Style Map Style Name are used.

For example a typeface named "GreatFace" with 10 fonts:

Id	Font	Font Family	Font Subfamily	Typographic Family	Typographic Subfamily	Width	Weight
1	GreatFace Narrow	GreatFace Narrow	Regular	GreatFace	Narrow	Condensed	400 - Normal
2	GreatFace Narrow Italic	GreatFace Narrow	Italic	GreatFace	Narrow Italic	Condensed	400 - Normal
3	GreatFace Light	GreatFace Light	Regular	GreatFace	Light	Medium	300 - Light
4	GreatFace Light Italic	GreatFace Light	Italic	GreatFace	Light Italic	Medium	300 - Light
5	GreatFace Regular	GreatFace	Regular			Medium	400 - Normal
6	GreatFace Italic	GreatFace	Italic			Medium	400 - Normal
7	GreatFace Bold	GreatFace	Bold			Medium	700 - Bold
8	GreatFace Bold Italic	GreatFace	Bold Italic			Medium	700 - Bold
9	GreatFace Black	GreatFace Black	Regular	GreatFace	Black	Medium	900 - Black
10	GreatFace Black Italic	GreatFace Black	Italic	GreatFace	Black Italic	Medium	900 - Black

Font	Bold	Italic	Italic Angle	PANOSE Weight	PANOSE Proportion
GreatFace Narrow				5 or 6 (Book or Medium)	6 - Condensed
GreatFace Narrow Italic		YES	12	5 or 6 (Book or Medium)	6 - Condensed
GreatFace Light				3 - Light	2, 3, or 4
GreatFace Light Italic		YES	12	3 - Light	2, 3, or 4
GreatFace Regular				5 or 6 (Book or Medium)	2, 3, or 4
GreatFace Italic		YES	12	5 or 6 (Book or Medium)	2, 3, or 4
GreatFace Bold	YES			8 - Bold	2, 3, or 4
GreatFace Bold Italic	YES	YES	12	8 - Bold	2, 3, or 4
GreatFace Black	Y/N			10 - Black	2, 3, or 4
GreatFace Black Italic	Y/N	YES	12	10 - Black	2, 3, or 4

All these settings and flags can be modified through the settings on the Font Properties panel.

Masters Tab:

- Italic Angle (0 for Upright, negative for Italic)

Instances Tab:

- Style Name
- Weight Class
- Width Class
- Typographic Family Name *
- Typographic Subfamily Name *
- Style Map Family Name *
- Style Map Style Name *
- PANOSE (optionally Weight and Proportion)

*) FontCreator can generate the additional naming fields, so you don't have to worry about them.

3.11 Font Embedding

From the **Font Properties** panel select the **Font** tab. Here you can alter the [Embedding Licensing Rights](#).

Note: Embedding (legacy) symbol fonts may not be possible in Word.

3.12 Monospaced versus Proportional

A monospaced font is a font where all characters have the same width. These fonts are often used to emulate typewriter output for reports, tabular work and technical documentation, but are also common in code editors.

In a proportional font the width of each character, including the space character, varies with the shape of the character. Proportional fonts are easier to read and are preferred for publishing applications.

From proportional to monospaced

To change a proportional spaced font into a monospaced font, follow these steps:

- Select the **AutoMetrics** command (**Tools** menu) to force the advance width to be the same for all glyphs including .notdef and combining marks. The only exception is (format) control characters (e.g. .null, zero width non-joiner, right-to-left mark) which are allowed to be zero-width.
- The advance width of combining marks may be collapsed through OpenType glyph positioning (e.g. using an OpenType single adjustment lookup in the mark feature).
- If necessary, change the outlines of glyphs that are too wide.
- In the **Font Properties** panel on the **Instances** tab set **Panose Family Kind** to 2 (Latin Text) and **Panose Proportion** to 9 (Monospaced).

3.13 Unicode versus Symbol

Important: Symbol fonts (as defined by the OpenType specification using cmap platform 3 and encoding 0) are legacy; we recommend designing a Unicode font instead. If there are code-points assigned for your characters, then use them, otherwise use the private use area.

Symbol character sets have a special meaning: all of the characters in the Unicode range 0xF000 - 0xFFFF (inclusive) will be used to enumerate the symbol character set. All glyphs in this range are mapped to the range 0x0000 - 0x00FF.

Symbol fonts do not form words, so line breaks can occur after any character code. A spell checker should not check symbol font-formatted material.

You can convert your font between Unicode and Symbol through the **Convert Font** menu item in the **Tools** menu.

Note: only the first 224 characters of symbol fonts will be accessible: a space and up to 223 printing characters.

3.14 Recommended Glyphs

In addition to script and language specific punctuation and native numbers, the following glyphs are highly recommended for inclusion in fonts.

.notdef - the very first glyph!

All fonts must include a .notdef (missing character) glyph as first glyph (glyph index 0). The .notdef glyph is very important for providing the user feedback that a glyph is not found in the font. For example, if your font doesn't contain the at sign (@), the user will see the .notdef glyph to warn the user about the fact the actual character is missing.

Note: sometimes an operating system, a word processor, web browser, etc. will use a [fallback font](#), so the user will still see characters that are not included in the actual font.

This glyph should not be left without an outline as the user will only see what looks like a space if a glyph is missing and will not be aware of the active font's limitation. It is recommended that the shape of the .notdef glyph be either an empty rectangle, a rectangle with a question mark inside of it, or a rectangle with an X. Creative shapes, like swirls or other symbols, may not be recognized by users as indicating that a glyph is missing from the font and is not being displayed at that location.



Note: since the .notdef glyph is important, there is an option that will ensure it is always included as first glyph in your exported fonts. See the [options dialog](#).

Obsolete glyphs - .null and nonmarkingreturn

The .null and nonmarkingreturn glyphs used to be required, but they no longer need to be included in your fonts. If you include them, it is best to follow these guidelines:

.null -> glyph index 1; no contours; zero advance width.

nonmarkingreturn -> glyph index 2; no contours; advance width equals the advance width of the space glyph.

Note: there is one exception: COLR version 0 color fonts should implement glyph index 1 as the .null glyph, as some early Windows implementations of the COLR table require it.

Space

Obviously the space character is very important. It usually comes right after the above mentioned glyphs, but its position is no longer relevant. The space glyph is often mapped to both space and no-break space; it has no contours and positive advance width.

General punctuation and Latin numbers

Glyph Name	Descriptive Name	Sample	Unicode
space	space		\$0020
exclam	exclamation mark	!	\$0021
quotedbl	quotation mark	"	\$0022
numbersign	number sign	#	\$0023
dollar	dollar sign	\$	\$0024
percent	percentsign	%	\$0025

Glyph Name	Descriptive Name	Sample	Unicode
ampersand	ampersand	&	\$0026
quotesingle	apostrophe	'	\$0027
parenleft	left parenthesis	(\$0028
parenright	right parenthesis)	\$0029
asterisk	asterisk	*	\$002A
plus	plus sign	+	\$002B
comma	comma	,	\$002C
hyphen	hyphen-minus	-	\$002D
period	period	.	\$002E
slash	slash	/	\$002F
zero	digit zero	0	\$0030
one	digit one	1	\$0031
two	digit two	2	\$0032
three	digit three	3	\$0033
four	digit four	4	\$0034
five	digit five	5	\$0035
six	digit six	6	\$0036
seven	digit seven	7	\$0037
eight	digit eight	8	\$0038
nine	digit nine	9	\$0039
colon	colon	:	\$003A
semicolon	semicolon	;	\$003B
less	less-than sign	<	\$003C

Quotes and Ellipsis

Smart quotes (also known as curly quotes) are fancy characters which make text look better compared to the straight apostrophe (') and straight quote or inches character ("). Microsoft Word also automatically changes 3 periods to an ellipsis.

If your font does not support these characters, you can either turn the "Smart Quotes" and "Ellipsis" features off in the application (In Microsoft Word it's under Tools -> AutoCorrect) or make sure these glyphs and their mappings are available in the font.

Glyph Name	Descriptive Name	Sample	Unicode
quotelleft	left single quotation mark	'	\$2018
quoteright	right single quotation mark	'	\$2019
quotedblleft	left double quotation mark	"	\$201C
quotedblright	right double quotation mark	"	\$201D
ellipsis	horizontal ellipsis	...	\$2026

Other suggested glyphs

Glyph Name	Descriptive Name	Sample	Unicode
nbsp *	no-break space		\$00A0
currency	currency sign	¤	\$00A4
section	section sign	§	\$00A7
logicalnot	not sign	¬	\$00AC
degree	degree sign	°	\$00B0
paragraph	paragraph sign	¶	\$00B6
periodcentered	middle dot	•	\$00B7

Glyph Name	Descriptive Name	Sample	Unicode
endash	en dash	–	\$2013
emdash	em dash	—	\$2014
bullet	bullet	•	\$2022
euro	euro currency symbol	€	\$20AC

* nspace is usually mapped to the space glyph

Additional glyphs for complex script fonts

Glyph Name	Descriptive Name	Sample	Unicode
zerowidthspace	Zero width space		\$200B
zwnj	Zero width non-joiner		\$200C
zwj	Zero width joiner		\$200D
dottedcircle	Dotted circle	◦	\$25CC
multiply	Multiplication sign	×	\$00D7
figuredash	Figure dash	–	\$2012
horizontalbar	Horizontal bar	—	\$2015
whitemediumsquare	White medium square	□	\$25FB
blackmediumsquare	Black medium square	■	\$25FC
whitemediumsmallsquare	White medium small square	◻	\$25FD
blackmediumsmallsquare	Black medium small square	◼	\$25FE

3.15 Last Resort Fonts

When an operating system (or software that comes with its own text rendering) needs to show text, it has to determine if the characters are available in the selected

font. If a character is missing it will either show the .notdef glyph, or it uses a fallback mechanism e.g. it will look for the character in other fonts. A last resort font ensures at least one font is able to show such missing character, those fonts are also known as **fallback fonts**.

Character to glyph index mappings

Typically last resort fonts contain a glyph for a specific character group, where such glyph represents the specific range of characters. This will ensure the number of glyphs remain low, while the font can contain mappings to all Unicode planes covering over one million code points. Within the OpenType specification there are basically two formats which can store many character to glyph index mappings. The common format which works great for regular fonts and a compact format which is optimized for storing ranges of characters mapped to single glyphs.

Windows doesn't support last resort fonts that make use of the compact format. Apple's Last Resort font uses the compact format. It is pre-installed on a Mac and contains all Unicode characters. The file size of the Last Resort font is 3KB, but if saved with the common format it will become 12.7MB.

By default FontCreator saves fonts with the common format, unless you check the "last resort font" box.

Blank Fonts

Some web sites use a blank font to detect whether a font is present on the system of the visitor. Other sites use such font to ensure no characters are shown from another font.

3.16 Single Line Fonts

True single line OpenType fonts don't exist, as OpenType fonts are based on the principle of closed contours. Basically the area inside a contour (or between two contours) is filled. In general these fonts are not designed for engraving or laser cutting, but with some tricks you can actually make fonts that **might** work for this specific task.

To help making such fonts, FontCreator allows you to design your glyphs with open contours. After enabling this Open Contour feature, FontCreator will show contours as single stroke outlines.

Do follow these steps to get started with open contours:

Enable Open Contours

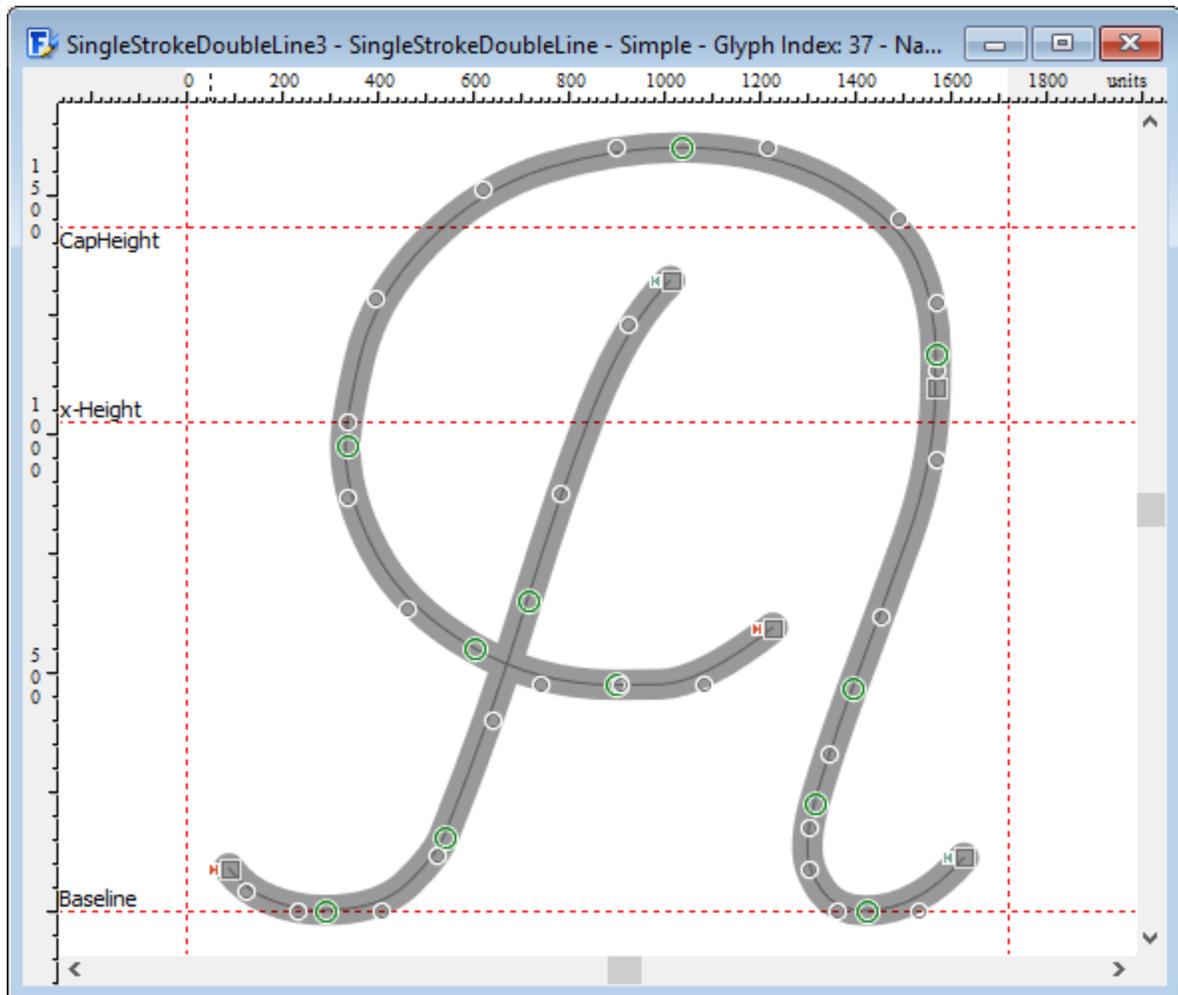
Go to the Options dialog, and select the [Font tab](#) to enable open contours. This will ensure the Open Contours option is always shown on the Font Export Settings dialog.

Set Font Export Setting for Open Contours

To enable open contour support for your font, go to the [Font Export Settings dialog](#), and set Open Contours to either Single Stroke or Double Stroke. The Double Stroke option is compatible with most engraving software, but the Single Stroke option is more efficient. We recommend to test which option works best in your situation.

Open a Closed Contour

After you've followed the above steps, you'll be able to open a contour within the Glyph panel. Select a contour, then right-click to select Toggle Open Contour. See [Contours](#) for more information.



Some contour related features will not work well with open contours. For example Optimize Contours will likely change the index of the first point, so either avoid it, or manually correct the first point. Right-click an on-curve point and then select First Point.

OpenType SVG with Open Contours

If you intend to make a single line font meant to be used within software that supports OpenType SVG fonts, you can colorize your glyphs, and then export the font with SVG outlines. SVG does support open contours, but the font also comes with the monochrome outlines with closed contours.

Tip: Open contours (and font export settings) will be saved within the font project file, so it is the preferred way to always work with font project files.

See also:

[Online Tutorial - Create OpenType Color Fonts](#)

Part



- [Empty glyphs](#)
- [Simple glyphs](#)
- [Composite glyphs](#)
- [Hybrid glyphs](#)

Tip: In the Glyph panel you can step through with the Back and Forward blue arrows in the Drawing toolbar to make adjustments.

4.2 Glyph Metrics

Several horizontal and vertical lines, also known as glyph metrics, will help you with your font design. You can activate them through the [Metrics Options](#) dialog.

Note: There are also global font metric settings. To change those font metrics, click Properties on the Font menu and then click the Metrics tab.

Left side-bearing, right side-bearing, advance width, top side-bearing, bottom side-bearing, and advance height

In the Glyph panel there are two vertical lines (normally the glyph outline lies in between these lines) that represent the left and right side-bearings. These are shown by default but you can hide them through the Show Metrics button on the Drawing toolbar. The left and right side-bearings can be changed by dragging them to their desired position. You can also adjust the side-bearings through the [Glyph Properties panel](#). White space should be evenly distributed between the left and right side-bearings of glyphs except when font is specifically designed with ligatures for joining characters in a cursive script font.

Non-spacing combining marks

These marks can be used to construct composite glyphs. [Complete composites](#) can help with this. Then they are usually also used in Mark-to-Base, Mark-to-Ligature, and Mark-to-Mark lookups in the OpenType Designer. Such OpenType layout features can be [automatically generated](#).

In general non-spacing combining marks should have a zero advance width. There is one exception as within a [monospaced font](#) all visible glyphs must have the same advance width.

Tabular figures

Tabular figures (i.e. the digits 0 - 9) should all have the same advance width.

Expressions

You can provide simple expressions to calculate the glyph metrics. These functions (as well as add, subtract, multiply and divide) are supported:

- | | |
|-------------------------------------|--|
| <code>min(X , Y)</code> | Returns the lesser value of the two input arguments X and Y. |
| <code>max(X , Y)</code> | Returns the greater value of the two input arguments X and Y. |
| <code>abs(X)</code> | Returns the absolute value of the argument X. |
| <code>lsb(glyphname, master)</code> | Returns the left side-bearing value for the glyph named glyphname. If you don't provide a glyphname, it assumes the current glyph. |
| <code>rsb(glyphname, master)</code> | Returns the right side-bearing value for the glyph named glyphname. If you don't provide a glyphname, it assumes the current glyph. |
| <code>aw(glyphname, master)</code> | Returns the advance width value for the glyph named glyphname. If you don't provide a glyphname, it assumes the current glyph. |
| <code>tsb(glyphname, master)</code> | Returns the top side-bearing value for the glyph named glyphname. If you don't provide a glyphname, it assumes the current glyph. |
| <code>bsb(glyphname, master)</code> | Returns the bottom side-bearing value for the glyph named glyphname. If you don't provide a glyphname, it assumes the current glyph. |
| <code>ah(glyphname, master)</code> | Returns the advance height value for the glyph named glyphname. If you don't provide a glyphname, it assumes the current glyph. |
| <code>bl(glyphname, master)</code> | Returns the bounding box left value for the glyph named glyphname. If you don't provide a glyphname, it assumes the current glyph. |

- `br(glyphname, master)` Returns the bounding box right value for the glyph named `glyphname`. If you don't provide a `glyphname`, it assumes the current glyph.
- `bw(glyphname, master)` Returns the bounding box width value for the glyph named `glyphname`. If you don't provide a `glyphname`, it assumes the current glyph.
- `bt(glyphname, master)` Returns the bounding box top value for the glyph named `glyphname`. If you don't provide a `glyphname`, it assumes the current glyph.
- `bb(glyphname, master)` Returns the bounding box bottom value for the glyph named `glyphname`. If you don't provide a `glyphname`, it assumes the current glyph.
- `bh(glyphname, master)` Returns the bounding box height value for the glyph named `glyphname`. If you don't provide a `glyphname`, it assumes the current glyph.
- `base(glyphname, master)` It will provide a value that syncs the specific metric (side-bearing, `aw`, or, `ah`) of the specified glyph. If you don't provide a `glyphname`, it assumes the current glyph for empty and simple glyphs, and it assumes the first glyph member for composite glyphs. This is especially useful for composite glyphs that have accents that go out of bounds of the base glyph. The function does not require a glyph name, which makes it even more convenient.
- `comp(glyphname, master)` Works with composite glyphs. It will provide a value that syncs the specific metric (side-bearing, `aw`, or, `ah`) of the specified glyph member. The expression requires a glyph name.
- `glyphname` To link glyph metrics, you can use a shorter notation for the side-bearing for the glyph named `glyphname`. If used within the expression for LSB it will process `lsb(glyphname)`, if used with RSB it will process `rsb(glyphname)`, and if used with AW it will process `aw(glyphname)`. This also works with TSB, BSB, and AH.

snap(snapname) This function can only be used with TSB and BSB. Snapname can be either typoascender, typodescender, xheight, capheight, baseline, or origin.

Glyph Names

To avoid conflicts, put the glyph name between double quotation marks if it contains ambiguous characters like hyphen, comma, parenleft, or parenright.

Master

This is the name of the master in a variable font. This parameter is optional. If possible avoid it, as it can make things rather complex.

Numbers

You can use numbers (use a decimal point with fractional numbers) and these functions along with the following operators: + - * / and use parentheses to define simple expressions. Here are some examples that illustrate the potential of expressions:

$\text{min}(18.75 * \text{aw}(\text{agrave}), 800) / 36 + \text{lsb}(\text{tonosmod-grek})$

$\text{lsb}(a) / 3 + \text{bw}(i) + 25 - (17.3 - 3) * 3$

Invalid Expressions

If an expression is invalid, the last known value will be used, and an exclamation icon will be shown on the right of the edit box.

Fixed Value

If you provide a fixed value expression, an equal sign followed by a value, it will keep the value, even when you move or resize glyph contours.

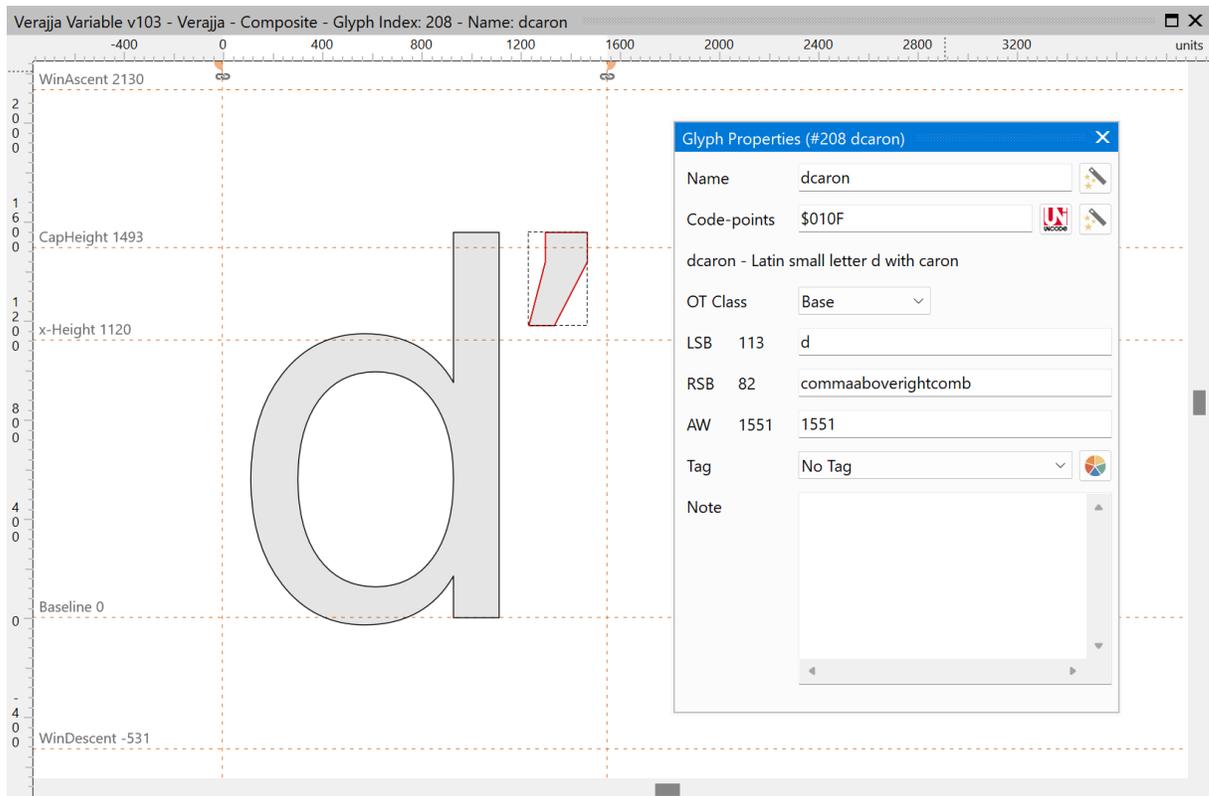
Link Glyph Metrics

Just type "n" in the LSB expression field for m if you want to keep the left side-bearing of glyph "m" and "n" to be the same.

Glyph Name Conflicts

If a glyph name is used in an expression, it could cause issues if it contains specific characters, e.g. a space character, or a character that is also used as an operator. Mainly the hyphen, which is both used to provide a script to the glyph name, but

also as an operator to subtract values. In such cases put the glyph name between double quotation marks.



Discarding Expressions

If LSB, RSB, and AW all have an expression, either RSB or AW will be considered invalid and will be discarded. Likewise with expressions for TSB, BSB, and AH, the one for BSB or AH will be discarded.

Auto Attach

With glyphs that have auto attach enabled, the metrics are automatically calculated based on the composite glyph members. Sometimes those metrics are almost perfect, but still need to be adjusted. In such case you can use expressions to add or subtract from the calculated values, by starting the expression with the equal sign followed by either a plus or a minus sign:

$+ = 10$ will add an additional 10 units to the calculated value.

$- = \text{aw}(\text{zero})$ will subtract the advance width of the glyph named zero.

Cancel Expressions

To stop using an expression, just type the number sign # and it returns to the last known value. Expressions will also be canceled when you enable auto attach, drag a side-bearing line within the Glyph Edit panel, or perform auto metrics or complete composites.

Shortcut for Default Bearings

Within the Glyph Edit panel press key ; to set default bearings.

Recursive Expressions

A recursive expression is one that uses a function which relies on itself. For example this combination:

LSB for glyph A = lsb(B) + 10

LSB for glyph B = lsb(A) + 25

Avoid recursive expressions, as such calculated values might vary unexpectedly.

Note: FontCreator takes the Italic Angle or the slant axis location into account while calculating metrics for horizontal writing. It also adds Caret Offset to the metrics, which if set correctly improves the visual layout of these metrics.

Note: Within a font with TrueType based outlines, composite glyphs may make use of a specific flag, known as the Use this glyph's metrics flag. If set, this forces the position of both left side-bearing and right side-bearing lines to match those from the glyph member. FontCreator will automatically set this flag when appropriate, but this option can be disabled through the [Options dialog](#).

See also:

[Metrics Options](#)

4.3 Empty Glyphs

Empty glyphs (like the space glyph) don't have outlines but they do have an advance width. In a **Glyph** panel you can change an empty glyph into a simple glyph by adding contours. You can also change an empty glyph into a composite glyph by adding **Composite Glyph Members**.

Make Empty

Select **Make Empty** from the Edit menu, or the Make Empty toolbar button from the Tools toolbar to delete the outlines from a glyph. This will also remove any color data if performed from the Font panel. If this action is performed from within a glyph panel, then only the outline for the active layer will be deleted.

4.4 Simple Glyphs

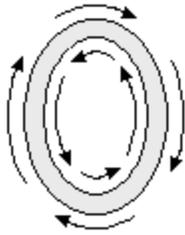
4.4.1 Introduction

A simple glyph consists of a series of contours. Contours are composed of straight lines and curves. Straight lines are defined by two consecutive on-curve points. Curves are defined by a series of points that are either cubic or quadratic Bézier curves.

TrueType outlines use quadratic (2nd-order) Bézier curves while CFF (also known as PostScript or Type 1) outlines use cubic (3rd order) Bézier curves. A curve always starts and ends with an on-curve point. A quadratic curve has one off-curve (control) point between start and end point, while a cubic curve has two off-curve (control) points. Such off-curve points are also known as Bézier control points (BCP).

However, with TrueType-based outlines there is an exception to this rule: if an on-curve point is exactly between two off-curve points, the on-curve point may be removed, as it will then become an imaginary on-curve point. Therefore any combination of off-curve and on-curve points is acceptable when defining a TrueType based outline.

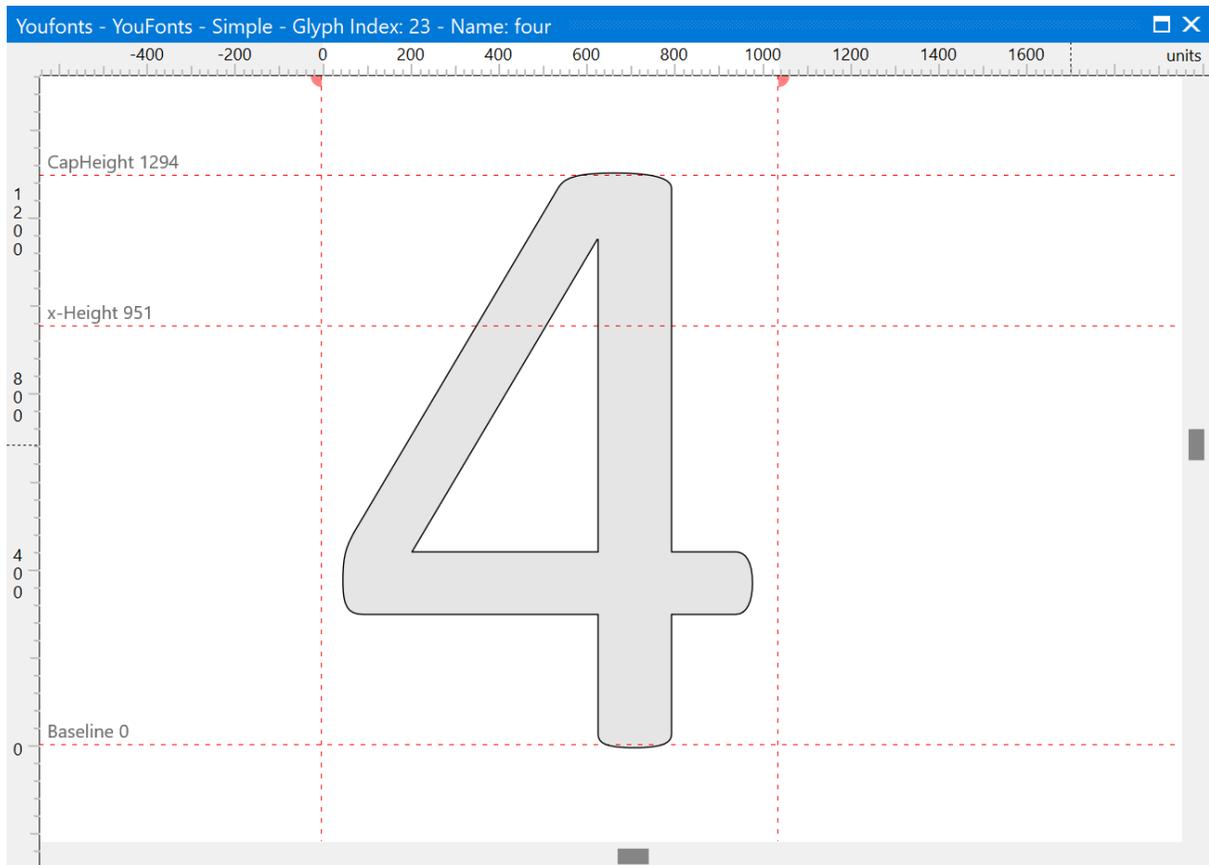
To distinguish between contour and point related operations, you can choose to work in contour or point mode. At any time you can change between Contour and Point mode; select the appropriate **Mode** from the **View** menu, press one of the triangle buttons on the **Drawing** toolbar or double-click anywhere in the glyph panel.



All contours have a direction. For **TrueType based contours**, thus the ones which use **quadratic Bézier curves**, the proper direction is explained here:

Contours that need to be filled black must have a **clockwise direction**. If we want to make a white area inside an existing contour we must make the direction of the new contour counter clockwise. Contour direction is determined by seeing in which direction the point index values increase or decrease. Contour direction is from the smaller point index to the larger. The general rule is that the contour direction should be such that "black is on the right". Using the glyph "O" as an example, the outer contour should travel clockwise, and the inner contour counter clockwise.

The character "4" is represented by a glyph with two contours. One contour you see as the black area and the white area within this glyph is the other contour.



For CFF based contours the direction is exactly the opposite, so contours that need to be filled black must have a counter clockwise direction.

These rules can be very confusing, but fortunately FontCreator can detect contours with a wrong direction. Click the Correct contour directions button on the Validation panel to correct the direction of all misoriented contours in a simple glyph or select Direction from the Edit menu to change the direction of the selected contour(s).

A hint will be shown in the upper left corner of a glyph panel if the glyph outline consist of a mix of quadratic and cubic based contours. The status bar will also show information about the contours.

From the Drawing toolbar you can change the way that you modify the glyph. In the Glyph panel, you can change between Contour and Point mode by double clicking inside the edit area, select the Mode from the View menu or use the appropriate button on the Drawing toolbar. The main difference between Contour mode and Point mode is that in Contour mode all operations are related to the contours while

in Point mode you can change parts of the contours (e.g. move, add and delete points).

Holding down the Shift key while dragging points or contours restricts movement so the selection moves only in the x or y direction. In the Glyph panel (in Point mode) the rectangles represent on-curve points and the circles off-curve points.

Tip You can nudge the selected contour(s) or point(s) up, down, left, or right by pressing the arrow keys. By holding down the Ctrl key and pressing the arrow keys finer nudging is available. By holding down the Shift key and pressing the arrow keys coarser nudging is available.

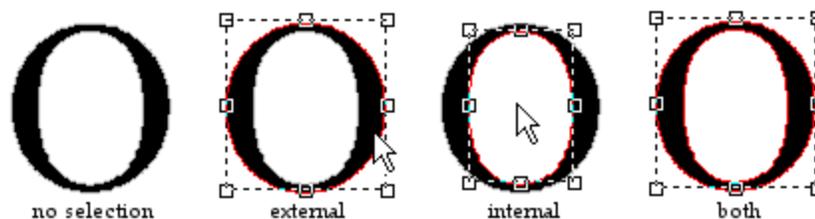
Note: The Validation features are not available in the Home Edition of FontCreator.

4.4.2 Contours

You can perform several operations on contours while in **Contour mode**.

Select contours

To select an external contour of a black area of a glyph click within the black area. To select an internal contour of a black area click within the internal white area.



To select more than one contour, press and hold down the Shift key while selecting contours. Another way is to click on the workspace where no contours are and, while holding down the left mouse button, drag a rectangle around all contours you want to select simultaneously.

Use the Ctrl-A shortcut or select **Select All** from the **Edit** menu to select all contours. Holding down the Shift key and clicking a contour already selected will remove that contour from the current selection.

Resize selected contours

When you select one or more contours, a box with “resizing handles” shows up around the selected contour(s). Click and drag one of those resizing handles to change the size of the selected contour(s). By default the selected contour(s) remain proportional to the original size, as you resize it. Press the Shift key while you drag one of the resizing handles located at the corners to resize freely.

Move selected contours

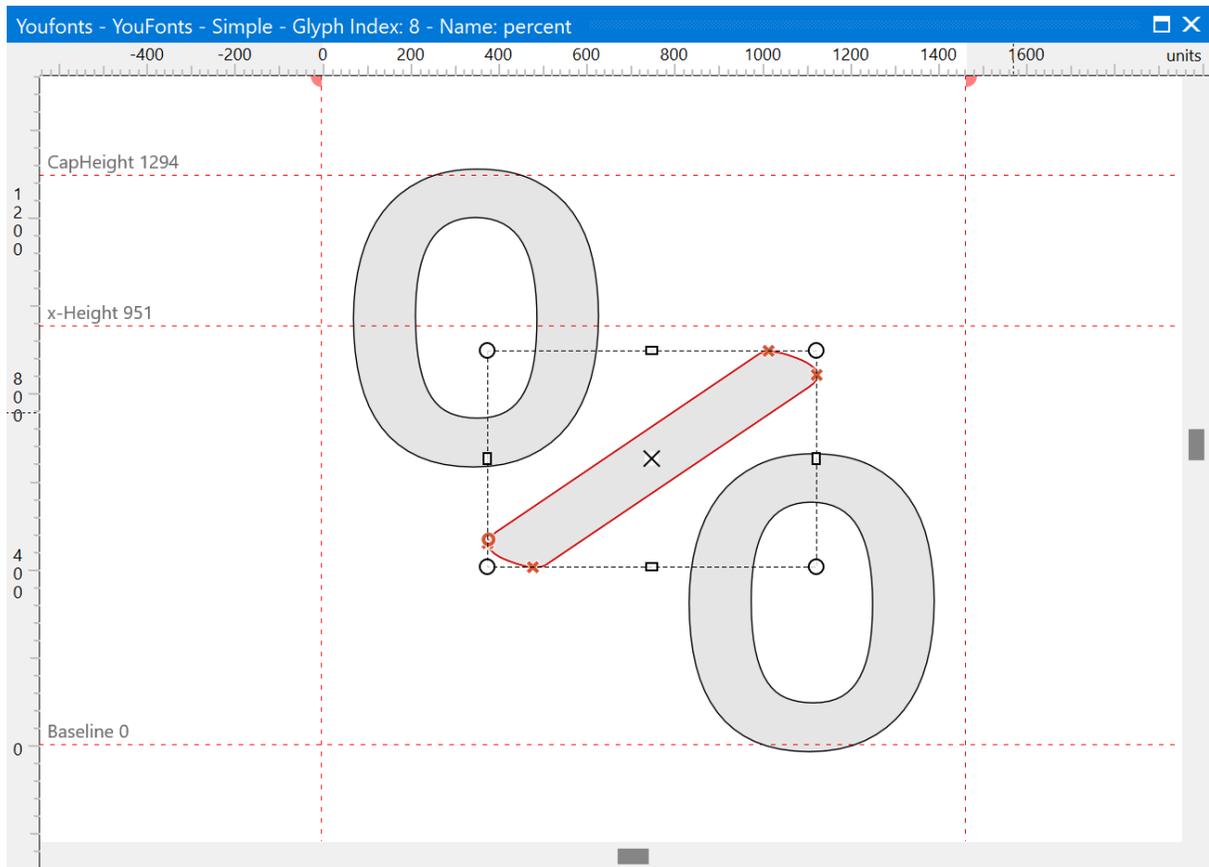
To constrain a selection so it moves only horizontally or vertically, press Shift as you drag the selection. Press Alt to ignore the snap to grid and snap to guidelines features.

Duplicate selected contours

To copy selected contours to a new position, hold down the Ctrl key as you drag the selection. Hold the Alt key to ignore the snap to grid and snap to guidelines features. Hold down the Shift key to move only in vertical or horizontal direction. Any combination of these keys is allowed.

Rotate and skew selected contours

You can rotate by first selecting a contour (or more) and then select it again (don't double-click). Alternatively press Key S to toggle between **Scale** and **Rotate and Skew** mode. The little rectangles on the corners will change into circles. These can be used to rotate the selected contour(s). The other four rectangles are used to skew the selection. To change the origin for the rotation you can move the little cross in the middle of your selection. To rotate by 15° steps, hold down the Shift key while rotating the selected contours. If Shift is held down on the skew operation, then it will skew by 1° steps.



Convert between Quadratic and Cubic Contours

As mentioned in the [simple glyphs introduction](#), a contour is either made out of **quadratic** or **cubic** Bézier curves. When you perform tasks like adding contours, FontCreator will add such contours as either quadratic or cubic, based on the TrueType/OpenType Outline Format setting in the Font Export Settings dialog.

If you wish to convert existing quadratic contours to cubic or vice versa, then select the contours, right-click to open a context menu from where you can select Convert.

Adding Contours and Outlines to a Glyph

Here are some ways to add a contour:

Rectangle and Square

To draw a rectangle, click the Rectangle toolbar icon on the **Drawing** toolbar. Then click within the Glyph panel to set the first corner of the rectangle. While keeping down the left mouse button, move to the opposite corner of the rectangle, and then release the left mouse button. If you want a perfect square, keep the Shift key down while defining the rectangle.

Ellipse and Circle

To draw an ellipse, click the ellipse toolbar icon on the **Drawing** toolbar and follow the same procedure as described with the rectangle above. Use the Shift key to make a perfect circle. Since outlines are made out of Bézier curves, and such curves can't be used to make a 100% perfect circle, FontCreator will use 12 points which is good enough for most situations. If drawing a circle or ellipse larger than the em square (units per em as defined on the General tab on the Font Properties dialog) it will appear with 20 (quadratic) or 24 (cubic) nodes to provide even more precision.

Insert a New Custom Contour

To insert a new contour, select Contour from the Insert menu. Click with the left mouse button to add points. You can create a curve if you hold down the mouse button while moving it around. With quadratic based contours you can also click the right mouse button to add an off-curve point. Creating points while holding down the Shift key generates straight lines (horizontal or vertical). After you have created your contour you press the Apply button.

Contour Direction

If you add a rectangle, square, ellipse, circle or a new custom contour, FontCreator will decide the contour direction, unless it is overlapping another contour. You can always change the direction of a contour if needed.

More Ways to Add Outlines (Contours) to a Glyph

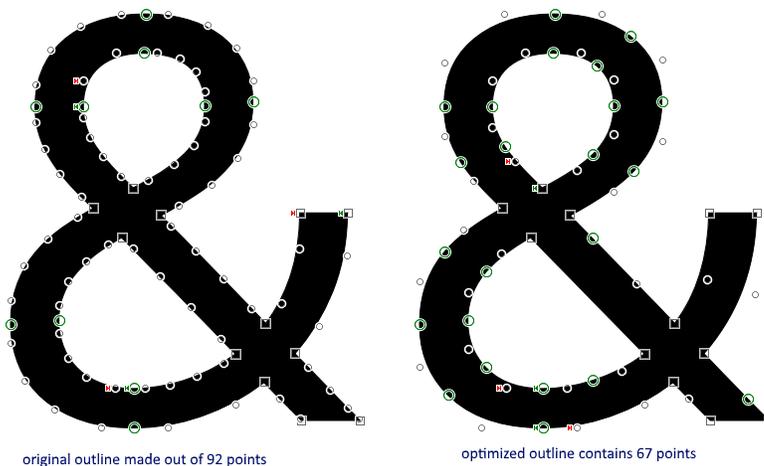
There are several other ways to add outlines (contours) to a glyph:

- Using the [Free Draw](#) Contours tool.
- Drag and drop a glyph from the **Samples** panel into a **Glyph** panel.

- Copy contours from other glyphs, even from other fonts and paste them. (These operations carried out in their respective **Font** or **Glyph** panels)
- Import a stored vector or bitmap file of an image of a glyph into a **Glyph** panel using **Import** in the **Tools** menu.
- Paste a Clipboard image of a glyph into the **Glyph** panel. The Clipboard image could be created in another application which has graphics editing capabilities e.g. a cropping function. This operation is performed using the **Paste** option in the **Edit** menu.

Optimize Contour

This advanced feature will try to reduce the number of points that make up the contour(s).



Note: the optimize contour feature might slightly change the original outline, so do experiment with it to ensure it suits your needs. At all times you can make use of undo and redo, so it should be easy to compare the optimized contours with the original outline.

Open Contours

The OpenType specification only allows for closed contours. To better assist with designing single line fonts for CNC engraving, we've added a special feature which allows you to work with open contours. This can be enabled from on the [Font tab](#) in the Options dialog.

In FontCreator open contours will only work with TrueType based outlines. Select a contour, then right-click to select Toggle Open Contour. The contour will be open if the first and last points of the contour are both on-curve.

If you want the contour to be open between two other adjacent points, select the one which comes right after the other (thus select the one which has a point index that is one more than the other) and right-click to select First Point.

See also:

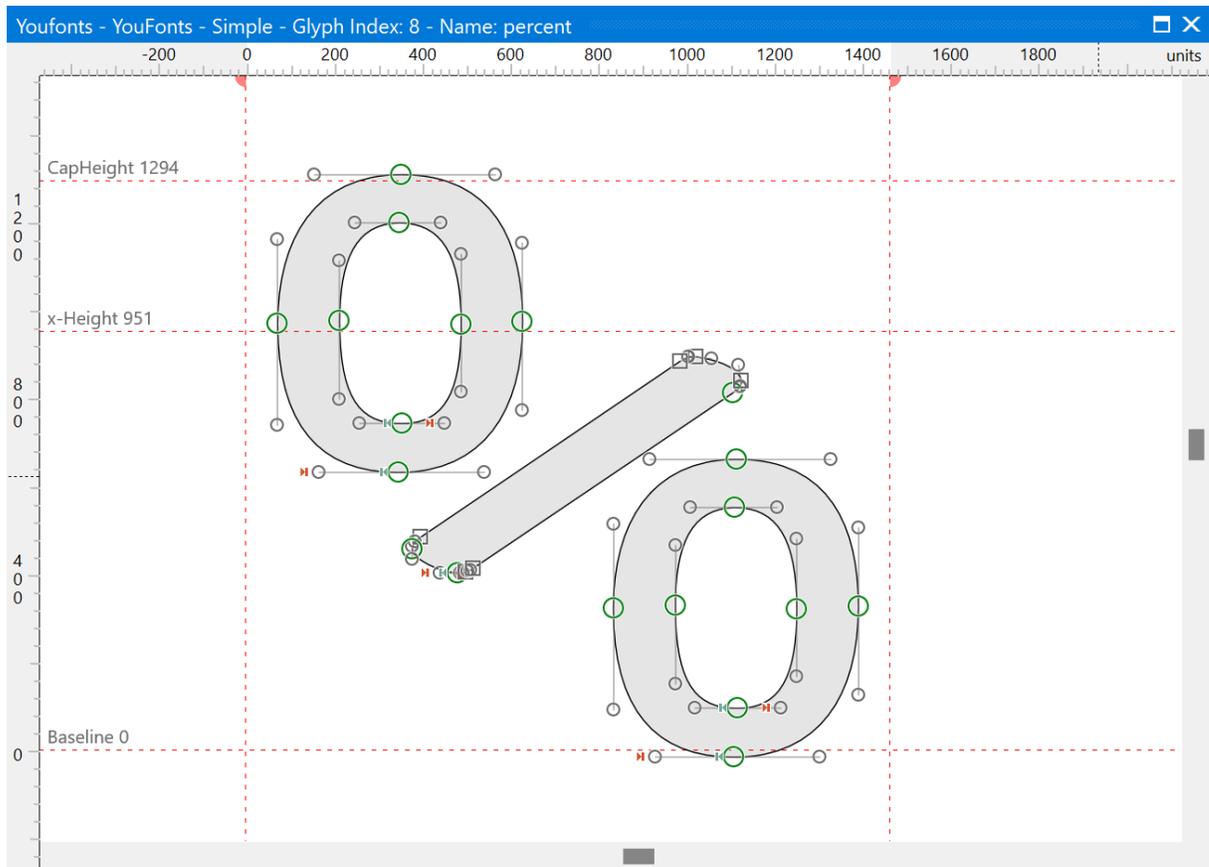
[Single Line Fonts](#)

4.4.3 Points

In **Point mode** the rectangles and the larger green circles represent on-curve points. The rectangles represent corners, while the large green circles indicate the curve is smooth. The smaller circles are off-curve points. Such off-curve points are also known as handles or control points.

There are several operations related to points (move, add, delete, change points to on or off-curve, etc.). Select one or more points and then right-click one of them to open a submenu with even more point related features.

Every contour has a start point with a green mark and an end point with a red mark. To change the start point, right-click a point and select **First Point**, or use the shortcut "Shift+F".



Change Curve Segments

To change a curve, just click on the curve segment, and move the mouse to the desired position.

If you want to divide a curve into two segments, just click on the curve.

Select Points

To select a single point click on it. To select more than one point hold down the Shift key while clicking on several points or another way is to hold down the left mouse button and drag a rectangle around the points you want to select simultaneously, whether a few or all points in the glyph. Use the Ctrl-A shortcut or select Select All from the Edit menu to select all points. Hold down the Shift key and select points you want to add or remove from the current selection.

The status bar shows context specific information, so in point mode it will show the mutual x and/or y position of the selected point(s) coordinate(s). Depending on the

number of points selected it might also show angle (with a single point) or distance (between two selected points).

Move Selected Points

You can select one or more points to move them around. To constrain a selection so it moves only horizontally or vertically, press Shift as you drag the selection. Press Alt to ignore the snap to grid and snap to guidelines features.

FontCreator will try to maintain smooth cubic curves, but you can override this by holding down Ctrl while moving selected points. FontCreator will then no longer keep smooth curves smooth, thus when you move a single off-curve point which used to be part of a smooth curve, then the opposite off-curve point will no longer move along.

There are some more advanced options which also only work with cubic curves:

While holding down Ctrl along with Shift, FontCreator will try to make all selected curves smooth.

While holding down Ctrl along with Alt, FontCreator will try to make all selected curves symmetrical.

Delete Selected Points

You can delete part of a contour by selecting one or more points and pressing the Delete key. If you are deleting part of a cubic curve, it is sometimes required to delete adjacent points to ensure that the remaining contour is still a valid cubic-based outline.

If you would like to keep the existing curvature, press the Backspace key, then FontCreator will try to keep the remaining contour as equal as possible to the original contour.

Scale Selected Points

You can scale by first selecting several points and then click on one of those selected points again (don't double-click). Alternatively press Key T to toggle between **Move** and **Scale** mode.

Smooth Curves

To guarantee smoothness, the on-curve point at which two curves meet must be on the line between the two off-curve points on either side. It is not always easy to keep an outline smooth around on-curve points. Fortunately there is a command that will do just that. Right-click one or more on-curve points, which are surrounded by off-curve points, and select **Smooth Curves**. If no points are selected it will apply to all points. This feature will only work if the on-curve point is already close to smooth.

Tip: There is also an access key "S" to smooth curves.

There is an additional **Smooth and Align Curves**, which in addition to smooth curves will also either vertically or horizontally align the related points, if they are already close to vertical or horizontal.

Tip: There is also an access key "D" to smooth curves and align points.

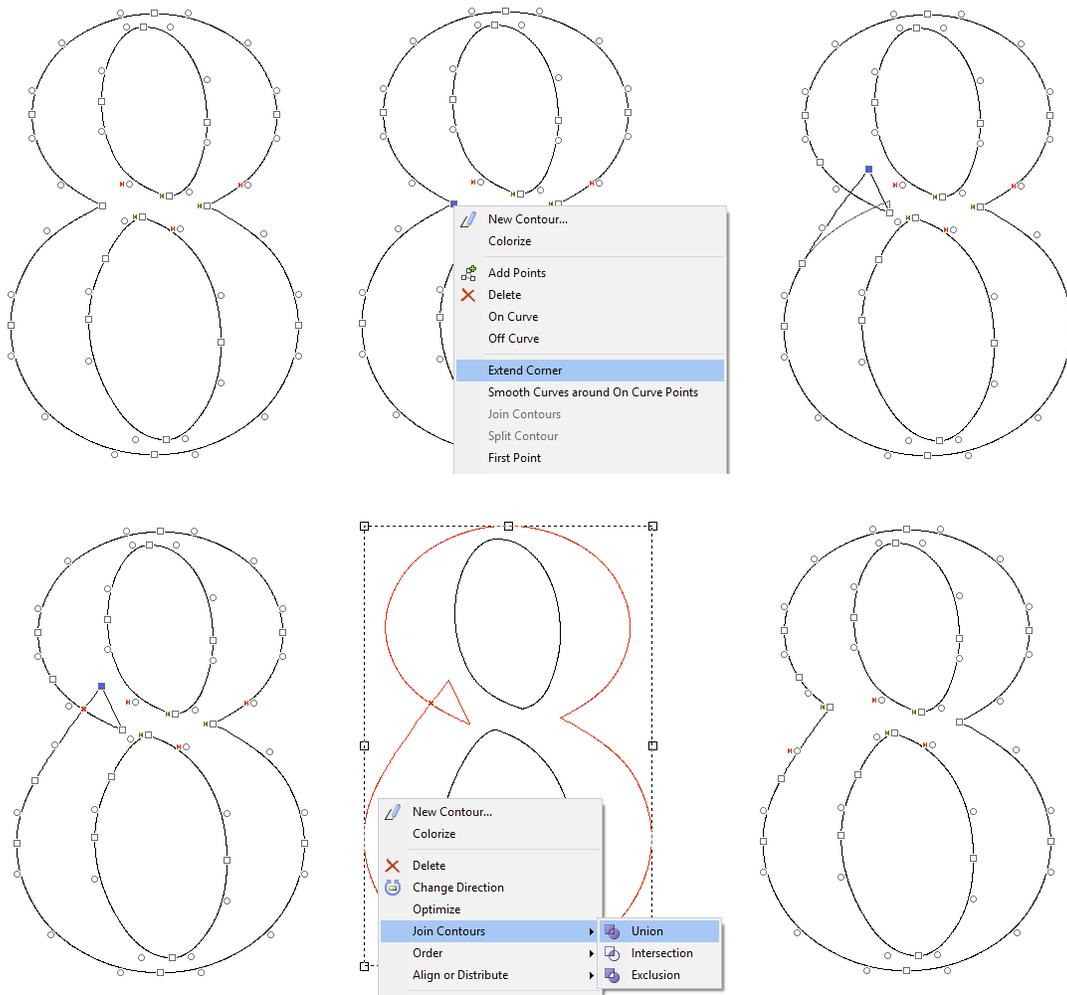
Round XY Coordinates

While editing glyph outlines the contours might have been moved or scaled which can result in points with where the X and/or Y value of the coordinate has become a floating-point number. On export FontCreator will always round to the nearest integer. Right click and select **Round XY Coordinates** if you want to round coordinates values while designing your outlines.

Extend Corner

The Extend Corner feature allows you to make changes to a contour segment without interfering with the other side of the corner. To use it, select one point, right-click and select **Extend Corner**.

The screen shots below show how a curve is changed with help of Extend Corner and Join Contours.



4.4.4 Join and Split Contours

Both Contour mode and Point mode have ways to combine and split contours. Usually Union, Intersection and Exclusion, available in Contour mode, and Knife, available in both modes, are recommended. If these features don't give expected results then Split Self Intersecting Contours available in Contour mode or Join Contours and Split Contour available in Points mode might help.

Union, Intersection and Exclusion in Contour mode

Use the **Union** feature to merge several overlapping contours.

Use the **Intersection** feature to keep all overlapping parts.

Use the **Exclusion** feature to remove all overlapping parts.

Using the knife tool in both modes

Use the **knife** tool to cut contours. Hold down the Shift key while using the knife tool to constrain the angle of cuts to 15° multiples.

Split Self Intersecting Contours

This feature is especially useful if you have a self intersecting contour that has a small part that is unwanted. This feature will split the contour, so you can then select the unwanted part, and remove it.

Join Contours and Split Contour in Point mode

When you want to combine two contours you have to select one point on each contour. Next right-click one of these points and click Join Contours on the shortcut menu.

To split a contour into two contours, select two points (these points should not be neighbors) on the same contour and then right-click one of these points and click **Split Contour** on the shortcut menu.

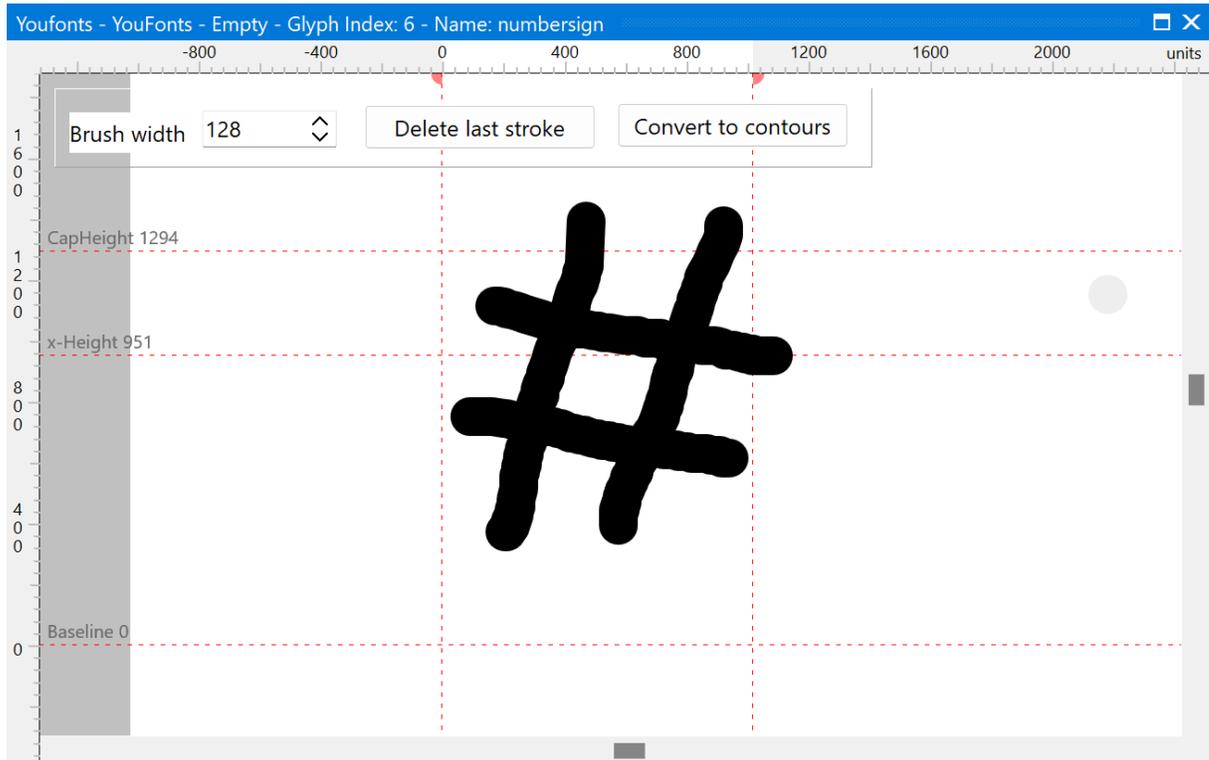
Note: Make sure the two combining contours have the same direction.

Note: **Union, Intersection, Exclusion and Knife** are not available in the Home Edition of FontCreator.

Note: If you apply **Union, Intersection, Exclusion or Knife** on a selection of contours that consist of both cubic and quadratic Bézier curves, FontCreator will first convert those contours to quadratic.

4.4.5 Free Draw Tool

The **Free Draw** tool can be used to manually draw lines that can be converted into contours. To enter **Free Draw** mode select **Free Draw Contours** from the **Insert** menu, or select the **Free Draw** tool from the **Drawing Toolbar**.



When the **Free Draw** tool is active, the Free Draw tool bar will be visible. On this tool bar you can set the Brush width, delete the last stroke or convert the drawn lines into contours.

To draw straight horizontal or vertical lines you can hold down the Shift key while drawing.

To erase part of the drawing, you can use the right mouse button as an eraser tool.

When you're satisfied, click on "Convert to contours" to add the contour to the currently displayed glyph (contours cannot be added to composite glyphs). To cancel, just close the Free Draw dialogue with the close button.

The new contour will not be joined to existing contours, even if it was drawn crossing them.

4.5 Composite Glyphs

4.5.1 Introduction

Composite glyphs are made out of “soft links” to one or more other glyphs - usually, but not limited to, a base character and one or more diacritical marks that are placed above and/or below the base character. Composite glyphs are supported by the TrueType based outline format, but will be decomposed on exporting a font with CFF based outlines.

Note: most Latin based composites can be generated through the [Complete Composites](#) feature.

Create composite glyphs

You can create a composite glyph when you are editing an empty glyph in a Glyph panel. To add a composite glyph member select Glyph from the Insert menu. To add one or more glyphs as composite glyph members first copy glyphs from the Font panel and then paste them into a Glyph panel that contains an empty glyph or a composite glyph.

Move selected glyph members

You can change the position of each of the used glyphs within the Glyph panel.

To constrain a selection so it moves only horizontally or vertically, press Shift as you drag the selection. Press Alt to ignore the snap to grid and snap to guidelines features.

Transformations

Besides moving glyph members, it is possible to perform operations like scale and rotation through the [Transform panel](#).

Duplicate selected glyph members

To copy selected composite glyph members to a new position, hold down the Ctrl key as you drag the selection. Hold the Alt key to ignore the snap to grid and snap to guidelines features. Hold down the Shift key to move only in vertical or horizontal direction. Any combination of these keys is allowed.

Replace a selected glyph member

You can easily replace a composite glyph member with another glyph. Within the Glyph panel right-click a glyph member, and click Replace.

Convert to simple glyph

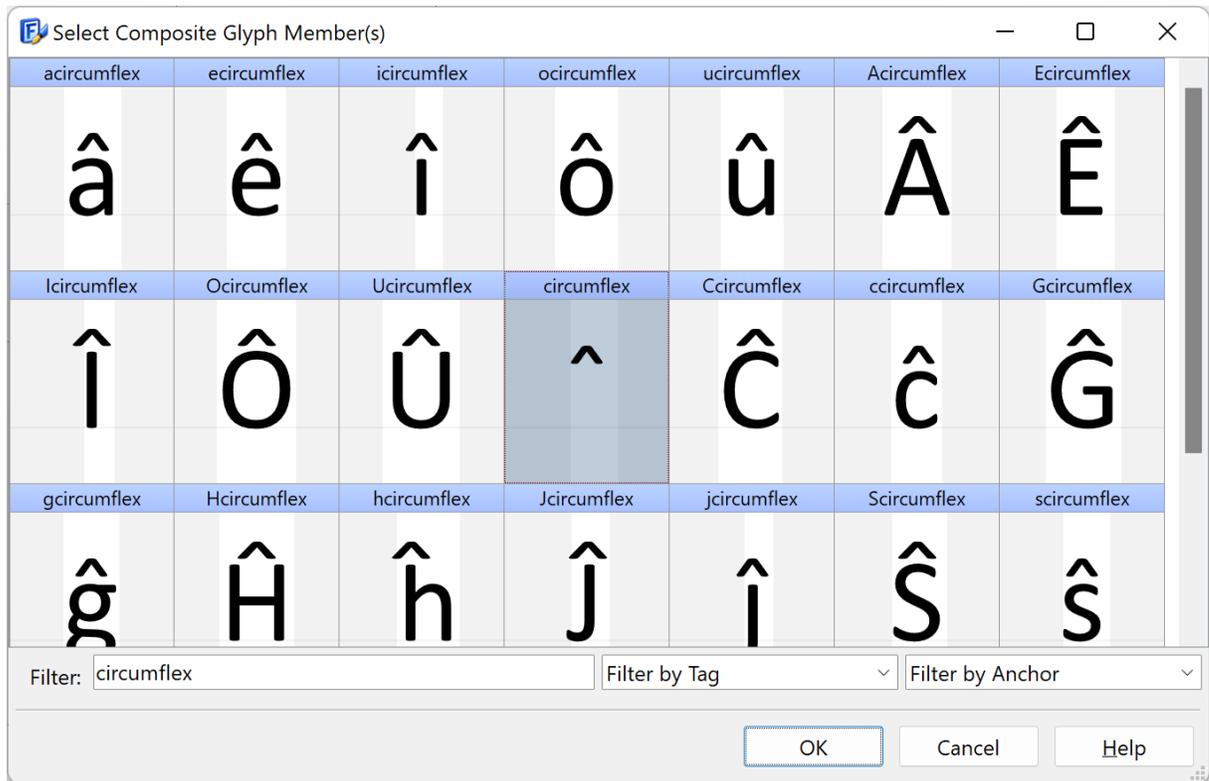
To convert a composite glyph to a simple glyph select the glyph in the Font panel or in the Glyph panel and select Make Simple from the Edit menu. Alternatively select one or more composite glyph members and right-click to select Decompose Components to only convert the selection to contours.

Join composite glyph members

If composite glyph members intersect such as with C cedilla, Get Union of Contours on the Glyph toolbar will convert the composite glyph to a simple glyph and join intersecting contours into one contour.

4.5.2 Add Glyph Member

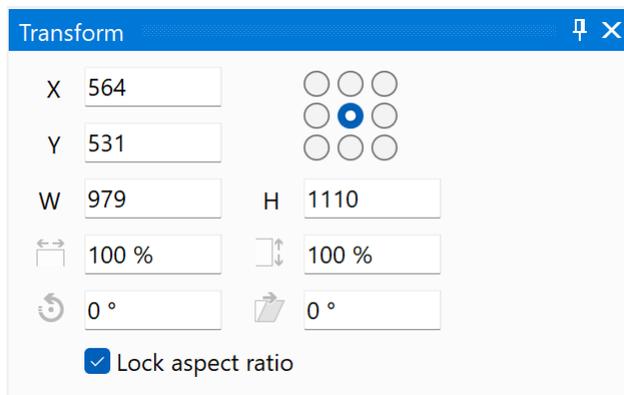
Select Glyph from the Insert menu to add one or more composite glyph members to a glyph.



In the Select Composite Glyph Member window, select the glyph(s) you want to add and press the OK button.

4.5.3 Glyph Member Properties

When you have opened a composite glyph in the **Glyph** panel, you can modify the properties of each composite glyph member through the Transform panel.



In the Transform panel you can modify the position, size, scale, rotation, and horizontal skew of the selected composite glyph members. In general it is best to avoid scaling, rotating, and skewing.

Note: There is an option to [decompose composite glyphs](#), on exporting a font, if one or more glyph members make use of scaled, rotated, or skewed properties.

4.5.4 Formula

A formula allows you to construct a glyph out of other glyphs. Here is an example:

`f_i_j=f+i+j`

It will add a glyph named `f_i_j` and will make a composite that is made out of these glyphs: `f`, `i`, and `j`.

The metrics of the composite glyph will also be calculated, and available kerning between individual glyphs will also be applied.

Composites and Anchors

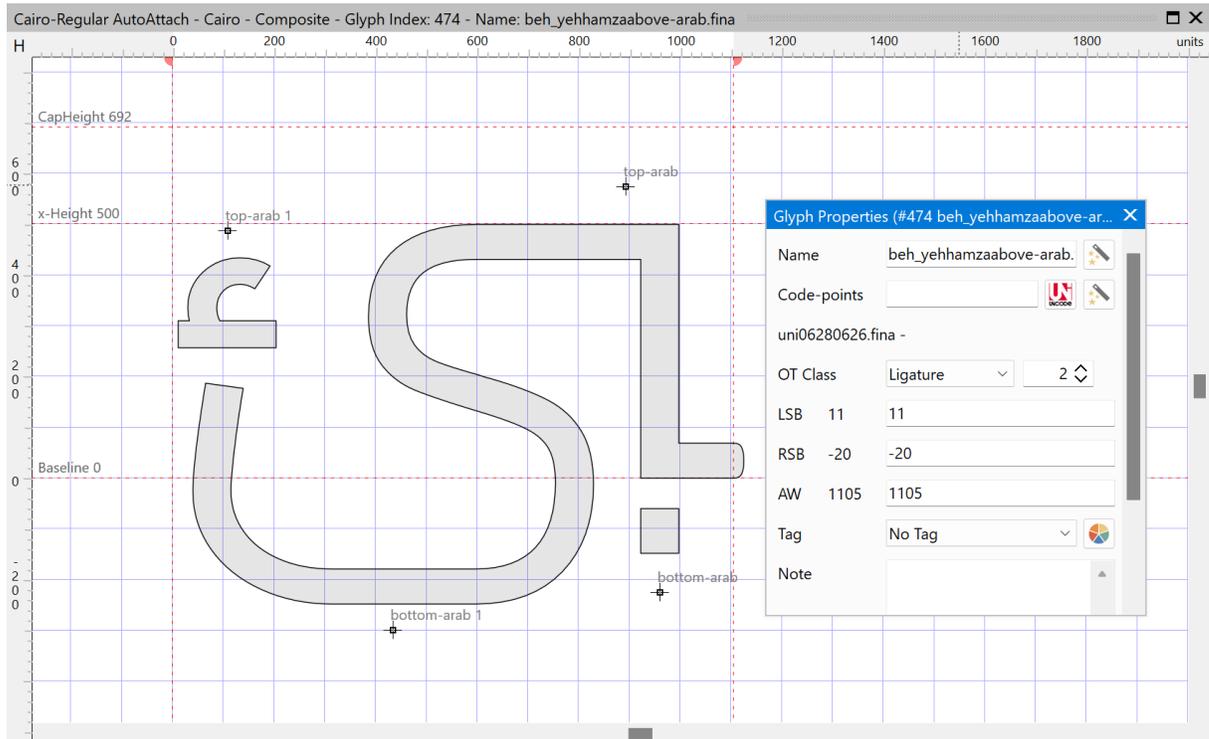
You can also make a pre-composed character. But to make this work, you first have to make sure the base and mark glyphs have corresponding anchors. Here is such formula:

`A+dotbelowcomb+brevecomb`

The `A` needs both a top and a bottom anchor, while the `dotbelowcomb` requires the mark bottom anchor and `brevecomb` the mark top anchor. The fastest way to add these [anchors](#), is to select the three glyphs and then go to the main menu to click Edit -> Complete Composites -> Anchor Based Reposition.

It can also be used to make more advanced composites like an Arabic ligature. For example:

`alefmaksura_alefmaksura-arab.fina+symboldotbelowmod-arab+arabichamzacom`



The created glyph uses a ligature glyph as base (`alefmaksura_alefmaksura-arab.fina`) and the two marks are added as ligature marks. The placement of the marks is also automated if the base and mark glyphs already contain the anchors. In this case the base glyph must also have the [OpenType Class](#) set to Ligature with 2 components.

4.5.5 Complete Composites

Use the Complete Composites feature to add composite glyph members to your glyphs. To use this powerful feature, select a glyph, or a range of glyphs, right-click and select Complete Composites. You can choose between these:

- Auto - uses a combination of Anchor Based and Composite Data as fallback
- Anchor Based - Uses anchors to create and position common composites that are made out of a base glyph and one or more combining marks

- Anchor Based Reposition - Same as Anchor Based, but will also reposition the anchors
- Composite Data - uses definitions from CompositeData.xml as explained below

Anchor Based

Using [anchors](#) allow you to build composites and to add OpenType features. Even though FontCreator automates most of this, there are some things crucial to make it work as expected.

If you want custom combining marks for capital letters, do give such glyphs a name with suffix `.case`, for example:

`gravecomb.case`

If you want to include custom marks for narrow glyphs, then add suffix `.narrow` to the marks, e.g.

`dieresiscomb.narrow`

If the font contains stacking marks, ensure the glyph names use the correct glyph name convention. For example, if you've created a glyph which consist of `circumflexcomb` and `gravecomb`, then name that glyph `circumflexcomb_gravecomb`.

The order of the glyph outlines/members within a stacking marks glyph is important, as the algorithm will position anchors based on the outlines of the first mark. So with `circumflexcomb_gravecomb.case` the first one should be the one that defines `circumflex`. To change the order, open the stacking mark in a Glyph panel, then select and right-click the specific mark, then click `Order -> Bring to Front`.

If you want to adjust the position of a mark on a base glyph, change the [anchor](#) on the base if you want to make the change for the whole group of marks. If you want to move the mark on all base glyphs, do change the anchor on the mark. After these changes, do select the composite glyphs that need to be repositioned and click `Complete Composites -> Anchor Based`.

If you want to revert to the automatic anchor positions, select the composites and click `Complete Composites -> Anchor Based Reposition`. Your manual adjustments to the specific anchors will be lost.

Note: FontCreator will position anchors based on the glyph outline. It will also take into account the [Italic Angle](#), so be sure it is set correctly.

Composite Data

The selected glyphs will be composed using data in CompositeData.xml, which is read when FontCreator first uses the feature. This feature works with over three thousand glyphs that are defined in this file. To get the most out of this feature follow these guidelines:

- The selected glyph(s) must be defined in the composite data file; either known by Unicode code-point or by glyph name.
- If the selected glyph(s) are empty or simple they will be replaced wherever composites are defined. Do not include simple glyphs in your selection if you don't wish to replace them with composites.
- If the selected glyph(s) are already composites they will be replaced only if the composite glyph members are different, not if their positions or scale factors are different.
- All composite members must be present in the font. If any composite members are mapped, but still empty, the composite glyph will be completed, though obviously missing the contours that have not been defined yet.

Info: Read the documentation about the content of the file CompositeData.xml and an explanation about how to modify and add glyphs. This document is available from our website:

<http://www.high-logic.com/font-editor/fontcreator/tutorials.html> 

Note: Complete Composites is not available in the Home Edition of FontCreator.

4.5.6 Auto Attach

If a composite glyph is made out of glyph members that all contain anchors meant to position the individual glyph members, then auto attach will do all the magic for you. Right-click one or more glyphs in the Font panel, and select Auto Attach -> Enable.

The order of the glyph members is important, as mark glyphs must come right after a base glyph. If the base glyph is not the first one, then select it, right-click, and select Order -> Bring to Front. Alternatively select Custom Formula from the Edit menu, to adjust the order of the glyph members through the [formula](#).

If your glyph is a ligature with marks, do ensure the Glyph OpenType Type as available on the [Glyph Properties](#) panel is set to Ligature and has the correct amount of ligature components.

4.6 Hybrid Glyphs

Since FontCreator 15 you can also mix contours and components in a single glyph outline. Such hybrid will be converted to a simple glyph on exporting to desktop or web font, as the OpenType font format does not support hybrid outlines.

4.7 Color Glyphs

4.7.1 Introduction

Each glyph can have its own color information assigned to it. If a host application does not support the color font extension, the regular outlines will be shown.

FontCreator allows you to add both COLR and SVG based color data as explained in the next topics.

Remove Color

To remove all color information from a glyph, right-click it in the font panel, and select **Decolorize**.

Switching Between Color Modes and Normal Mode

When editing a glyph, click the **Color Mode** button  or select **Color Mode** from the **View** menu.

See also:

[Color extensions](#)

4.7.2 Layered Color Glyph (COLR)

This topic is about adding color glyphs with the COLR extension. There are a couple of similarities with making a composite glyph, but there are also several differences.

Create a Single Color Glyph

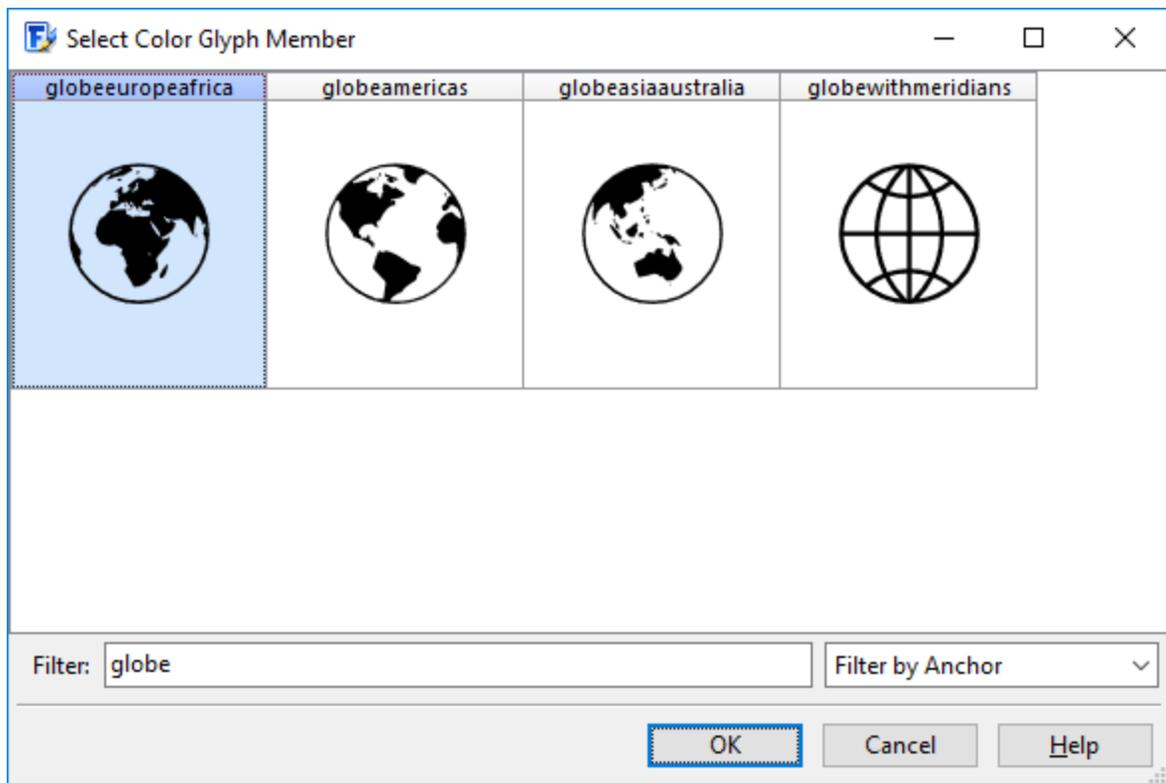
The easiest way to create a color version of a glyph is to select the **Colorize** option from the right-click menu when editing a glyph. This will automatically add the current glyph and assign a single color to it.

Create a Multi-color Glyph

To make a multi-color glyph, you need to add other glyphs and then define a color for each of these color members. Unlike composite glyph members, you can't change the size or position of these color members. You will need to make sure the individual glyphs are all designed to fit into the color glyph. One other important requirement is the fact that the metrics of the base glyph and the color members must be the same. So the offsets (usually set to 0) and the advance widths have to match.

You can use expressions to ensure the [glyph metrics](#) are correct for the color members.

To add one or more Color Glyph members, make sure you are in **Color COLR Mode** and right-click in the Glyph panel and select **Add** or click the Add icon **+**.



In the **Select Color Glyph Member** window select the glyphs you want to add to the color glyph and press the **OK** button.

You can change the order of the color glyph members via the up and down buttons on the [Color Glyph Member](#) panel, the **Order** options in the right-click menu or via the **Alignment** toolbar. To show and/or hide toolbars, select **Toolbars** on the **View** menu.

Note: for compatibility reasons the second glyph in OpenType COLR version 0 color fonts should be the .null glyph.

See also:

[Recommended Glyphs](#)

4.7.3 SVG based Color Glyph

This topic is about adding color glyphs with the SVG extension.

To enable the SVG options in the Glyph panel, ensure it is in Color SVG mode.

Generating an SVG color glyph

The easiest way to add a SVG color outline, is to generate it from the COLR outline. In Color SVG mode, right-click and select Color -> Convert COLR to SVG.

Alternatively you can import an SVG file. Be sure the transform used within the SVG document is adjusted so the color outline is positioned and sized correctly.

Within the glyph panel go into SVG color mode. Then right-click inside the edit area to edit or import the SVG document. Here is the content of one such SVG document:

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
<g transform="scale(1,-1)">
<path d="M701 1182c-334 0-605-271-605-605 0-334 271-605 605-605 335 0 606 271 606 605 0 334-271 605-606
605 z" fill="none" stroke="#FFF000" stroke-width="64" stroke-linecap="round" stroke-linejoin="round"/>
<path d="M886 903c57 0 103-31 103-69 0-38-46-69-103-69-57 0-103 31-103 69 0 38 46 69 103 69 z M520 903c57
0 103-31 103-69 0-38-46-69-103-69-57 0-103 31-103 69 0 38 46 69 103 69 z" fill="none" stroke="#00DDFF"
stroke-width="64" stroke-linecap="round" stroke-linejoin="round"/>
<path d="M1112 555c0-171-182-309-406-309-224 0-405 138-405 309" fill="none" stroke="#FF0000"
stroke-width="64" stroke-linecap="round" stroke-linejoin="round"/>
</g>
</svg>
```

Tip: if you design your color glyphs through the CPAL extension, you can let FontCreator generate SVG color data on [export of your font](#). So then there is no need to manually add SVG color data.

4.7.4 Palettes and Colors

A font can contain one or more palettes that allow a host application to quickly change the color scheme. By using palettes you can make different color schemes that allow your font to use different colors with, for example a different background color.

Color palettes and the COLR extension

This section describes how color palettes work with the COLR extension.

The palette colors are global, so if you change a color which is assigned to several color members, then all those members will use the updated color. If you want a

unique color for a specific color glyph member, then add a color to the palette and use that instead.

When in **Color Mode** you can assign colors to specific members, by selecting the **Paint Bucket** tool  from the **Drawing Toolbar** and click on one of the color glyph members in the Glyph panel. You can change the current color by clicking on one of the palette entries in the **Palette panel**.

The color palettes are not fully supported by all web browsers and software. Read more about it in our online color tutorial:

<https://www.high-logic.com/font-editor/fontcreator/tutorials>

See also:

[Palette](#)

Part



5 OpenType Layout Features

5.1 Introduction

OpenType Layout Features can be used to extend the functionality of fonts and create advanced typographic features and add additional capabilities such as ligatures, alternate glyphs, two-dimensional glyph positioning, specific functionality for different scripts and languages and much more.

When a font is opened, FontCreator will process all supported features in the font and split them up into two separate parts. The first part, the glyph substitutions, will be decompiled into a script that can be edited using the built-in script editor. The second part, the glyph positioning, will be processed to make them visually editable in the OpenType Designer.

If the font you're opening contains **Microsoft VOLT** project data for OpenType Layout Features, you will be asked if you want to convert this into a FontCreator project. If you choose not to use Microsoft VOLT project data, the binary OpenType Layout Features (GPOS, GSUB, and GDEF tables) will be used instead. You can also import Microsoft VOLT projects later through the [OpenType Designer](#).

Note: Microsoft VOLT project conversion and importing is only available in the Professional edition of FontCreator.

5.2 Types of substitution and positioning

The OpenType Layout Feature specification describes eight types of substitution lookups in the glyph substitution table (GSUB) which are all supported by FontCreator:

#	Type	Description
1	Single	Substitute a single glyph by another single glyph (a -> b)
2	Multiple	Substitute a single glyph by other multiple glyphs (a -> xyz)
3	Alternate	Substitute a single glyph by one of multiple alternates (a -> x or y or z)

#	Type	Description
4	Ligature	Substitute multiple glyphs by a single ligature (f f i -> ffi)
5	Context	Substitute one or more glyphs in context
6	Chaining Context	Substitute context specific glyphs (3rd -> 3 rd)
7	Extension Substitution	See * below
8	Reverse chaining context	Applied in reverse order, replace single glyph in chaining context

The OpenType Layout Feature specification describes nine types of positioning lookups in the glyph positioning table (GPOS) which are all supported by FontCreator:

#	Type	Description
1	Single adjustment	Change the position of a single glyph (sub/superscript)
2	Pair adjustment	Mostly used to define kerning pairs
3	Cursive attachment	Used for scripts that require glyphs to attach to the previous glyphs at exit and entry points
4	Mark-to-Base attachment	Attach a combining mark such as a diacritic to a base glyph
5	Mark-to-Ligature attachment	Attach a combining mark to a ligature
6	Mark-to-Mark attachment	Attach a combining mark to another mark
7	Context Positioning	Position one or more glyphs in context
8	Chained Context Positioning	Position one or more glyphs in chained context
9	Extension Substitution	See * below

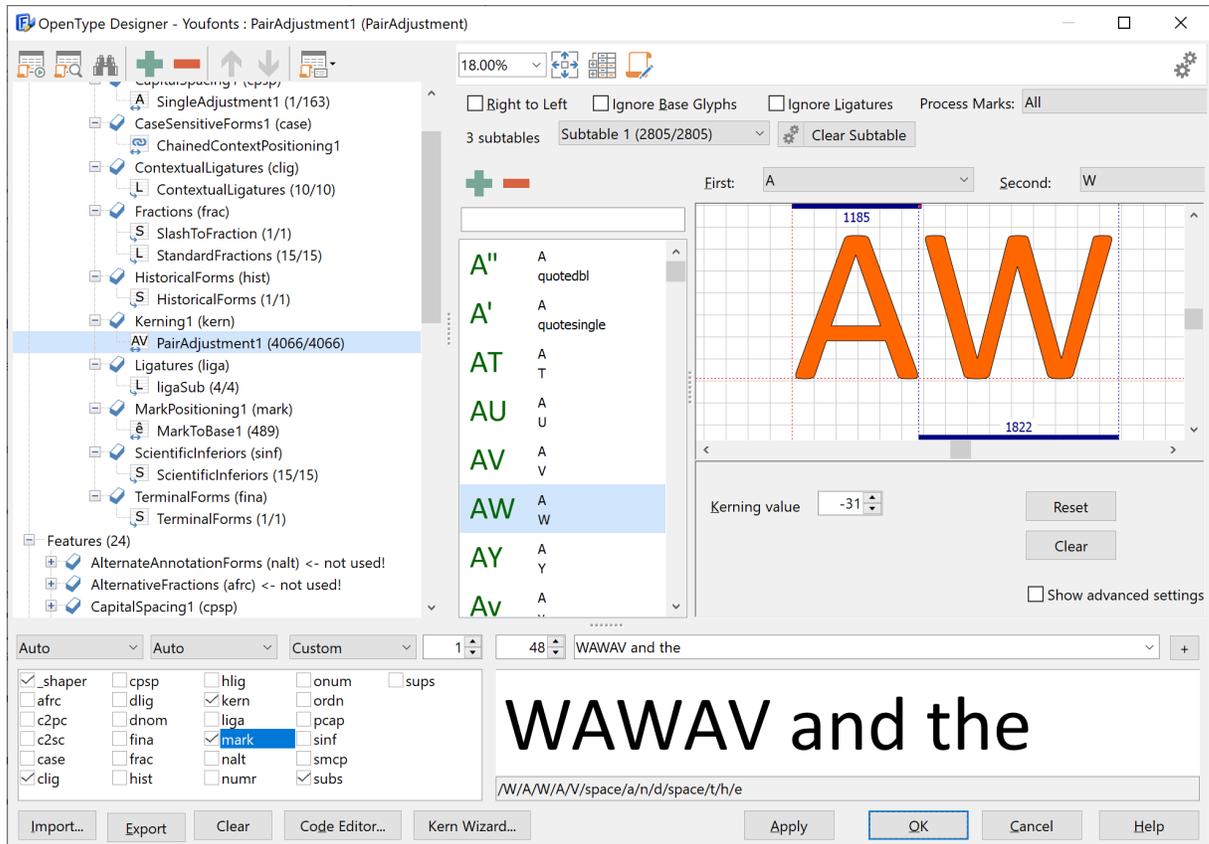
* Extension substitution is a special kind of lookup that is only used for fonts with a lot of features. FontCreator will automatically include such lookup if needed.

In addition FontCreator also supports all [feature parameters](#) currently defined.

So basically FontCreator supports all OpenType Layout Features. For an up-to-date list of things currently not fully supported, please visit our forums at <https://forum.high-logic.com/viewtopic.php?f=4&t=5098>

5.3 OpenType Designer

5.3.1 OpenType Designer



The OpenType Designer provides you with an easy to use visual way to edit glyph positionings. In the left pane you see how the features and lookups are organized in your font. The right pane allows you to edit the selected lookup. Some features (cv01-vc99, size, and ss01-ss20) have additional [feature parameters](#) that you can edit in the right pane.

The splitter on the dialogue can be dragged to adjust the width of the panes. Double-click the thumb bar (!) to dock the script pane when it is not needed or to undock it.

The left toolbar allows you to add, remove, move, rename and change scripts, languages, features and lookups. Most of these functions are also available in the right-click menu when you have selected an item.



Generate features	Automatically add numerous OpenType layout features to your font.
Proofing	Shows the proofing window which allows you to debug your OpenType layout features.
Explore	Shows the explorer window which allows you to easily find glyphs and glyph classes within your OpenType layout features.
Add	Add a new script, language, feature or lookup. The available options depend on which item is selected in the tree. If, for example, you select a feature, the current script and language will already be pre-determined on the add dialog.
Delete	Deletes a script, language, feature and/or lookup. A popup window will ask you to confirm and select how you want to delete the selected item.
Move up	*) Move a lookup up
Move down	*) Move a lookup down
Auto naming	This will provide all your scripts, languages, features and lookups with consistent and if possible meaningful names
Generate kerning	This will automatically create kerning pairs for pair adjustment lookups. See also Autokern .
Generate classes	Only affects Single and Pair Adjustment lookups: Will (re)group individual glyphs into glyph classes, based on their left and right side bearing, width and current adjustment value. This will greatly reduce the number of visible kerning pairs and allow you to quickly change adjustment values of multiple glyphs at once.
Break classes	Only affects Single and Pair Adjustment lookups: Will break all classes into separate kerning pairs or single adjustments.
Cleanup	Permanently deletes all unused scripts, languages, features, and lookups.

Lookup Order

*) The order in which lookups are defined is also the order in which they are processed by host applications. Some features (and thus also their lookups) are processed in a specific order, as described below in section **Shaping Engine**.

The right-click menu on the left pane also provides some additional lookup specific actions:

Rename Rename the selected item. Please note that each script, language, feature or lookup name has to be unique.

Change tag Allows you to change the tag (type) of scripts, languages and features. Please note that each tag may only exist once on the same level of the tree.

Autoclass Perform auto grouping on the selected lookup table only.

Break classes Break kerning classes into separate kerning pairs.

Preview Area

The preview area allows you to quickly test your OpenType layout features. The [Preview in FontCreator](#) section explains what input you can use in the Preview text and how the shaping engine works.

Jump to Processed Lookup

To see what feature was applied for a specific character or range of characters, click the glyph within the preview area. If the glyph is a result of a substitution, or positioning was applied to it, the last applied lookup will be shown above. This allows you to quickly inspect and fine-tune the OpenType layout features.

Import

The import function allows you to import an OpenType layout feature script. Both OTLFD and FEA syntax are supported. The Professional Edition of FontCreator also allows you to import **Microsoft VOLT** project (*.vtp) files.

Export

The export function allows the export of all scripts, languages, features, lookups, and class definitions.

Exchange and Reuse OpenType Features

Export from one font and then import into another font allows you to easily reuse OpenType features. If your destination font doesn't contain all source glyphs, set the "Ignore Unknown Glyphs" option, and the import feature will subset the OpenType features.

Clear

Removes all OpenType features. This includes **all** scripts, languages, features, feature params, lookups, classes, and anchors. Will be asked if you also want to drop anchor data. Keep in mind that anchor data is used by [Auto Attach](#) glyphs.

Lookupflags

Right to Left From a technical point of view, this is only important for Cursive Attachment lookups. When checked the last glyph in a given sequence to which the cursive attachment lookup is applied, will be positioned on the baseline.

For all other lookups it is used within FontCreator to indicate that the lookup is used for right to left writing. For kerning pairs this means that the first and second glyph will be visually swapped, which is recommended while working on characters that are used in right to left scripts like Arabic and Hebrew.

Ignore Base Glyphs If checked, the processing application will skip over the base glyphs*

Ignore Ligatures If checked, the processing application will skip over ligatures*

Ignore Marks If checked, the processing application will skip over marks*

Mark Filtering Set Defines which marks* should be filtered

***) Base, Ligature, and Mark are OT Classes which can be defined through the [Glyph Properties](#) panel.**

Clear subtable will delete all entries of the currently selected subtable, to clear an entire lookup table (including all subtables) use the right-click menu in the left pane.

Settings

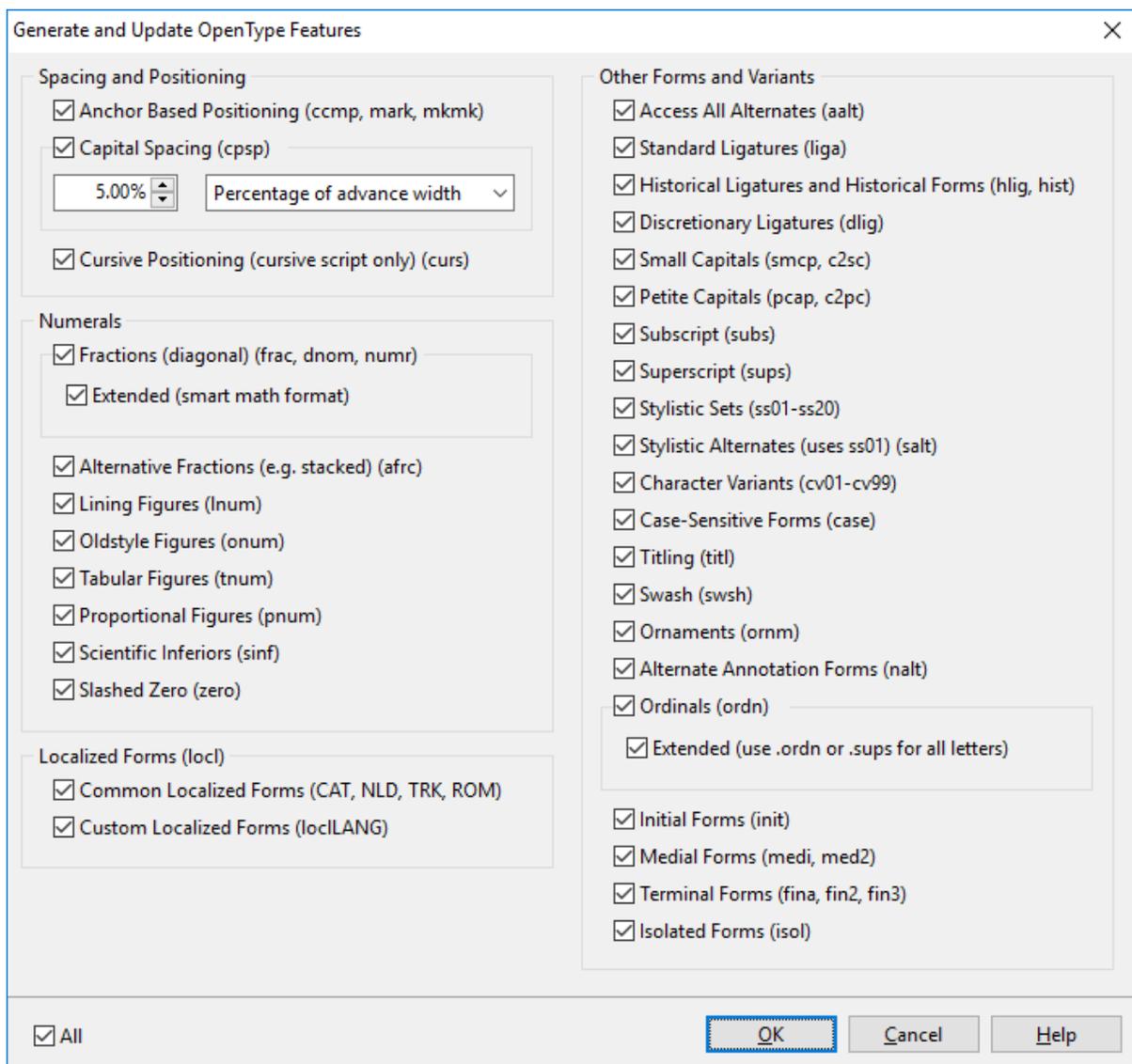
The settings button will open the [Designer Settings](#)

See also:

[OpenType Layout Feature Proofing](#)

5.3.2 Automatic OpenType Layout Features

Besides automatically creating [kerning pairs](#) for pair adjustment lookups, FontCreator can automatically add numerous other OpenType layout features to your font. Do ensure your glyphs have friendly name as explained [here](#). Within the OpenType Designer window click the first icon from the upper left corner. You will now be able to select which features you want to generate and/or update.



The generated lookups and classes have names that start with a script abbreviation. E.g. `latn_liga` for standard Latin based ligatures. When the features are updated, the existing lookups and classes will be deleted, so if you add custom features, better avoid mixing them with the generated features.

Note: Greek and Cyrillic characters are grouped together with Latin (`latn`).

Unless noted otherwise, the generation of features and lookups is limited to Latin, Greek, and Cyrillic scripts.

Anchor Based Positioning (ccmp, mark, mkmk)

Position mark glyphs with respect to mark and base glyphs and also compose and substitute several glyphs.

If your font contains glyphs with [anchors](#), then you can generate positioning features based on those anchors. Use [complete composites](#) to automatically generate anchors and build composites based on those anchors.

If you want to use different marks for upper and lower case letters, do add mark variants with a `.case` suffix. If you need special marks for narrow letters, like `i` and `j`, create additional mark glyphs with a `.narrow` suffix. If you want additional marks for small capitals, use `.smcp`; for petite capitals, use `.pcap`. If you want to use marks for both small capitals and petite capitals, add marks with a `.cap` suffix.

Capital Spacing (csp)

Globally adjusts inter-glyph spacing for all-capital text. Most typefaces contain capitals and lowercase characters, and the capitals are positioned to work with the lowercase. When capitals are used for words, they need more space between them for legibility and esthetics.

You can choose to increase the space by adding a uniform percentage of the advance width of each glyph (recommended per the specification), or a fixed value as indicated by a percentage of `upem`.

Cursive Positioning (curs)

In cursive scripts like Arabic, this feature cursively positions adjacent glyphs.

This feature is used in cursive scripts and will make use of entry and exit anchor points. For cursive scripts, ensure that each initial glyph has an exit anchor point, each medial glyph has both entry and exit anchor points, and each final glyph has an entry anchor point.

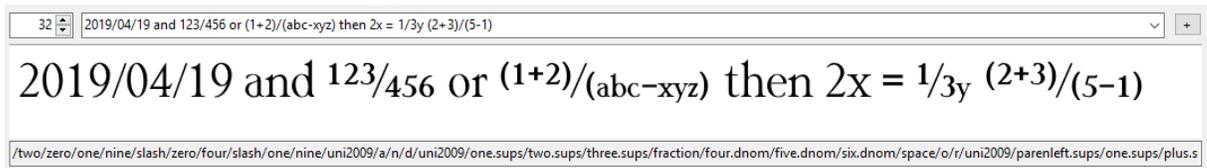
Fractions (diagonal) (frac)

Replaces figures separated by a slash with diagonal fractions. If the extended option is selected, it will generate lookups with contextual substitutions.

For this feature to work, these glyphs are required:

Digits (zero-nine), and optionally parenleft, parenright, hyphenminus, minus, plus, etc. as well as the variants for denominators with suffix .dnom, .subs, or inferior, and for numerators .numr, sups, or superior. At least the digits must have the same suffix, but in general it is best to use the same suffix for all denominators, and the same suffix for all numerators.

Also required are slash and fraction; optionally are space and thinspace.



If not all required glyphs are available, or if the extended option is not selected, it will create a ligature lookup with the following pre-composed fractions:

onehalf, zerothirds, onethird, twothirds, onequarter, threequarters, onefifth, twofifths, threefifths, fourfifths, onesixth, fivesixths, oneseventh, oneeighth, threeeighths, fiveeighths, seveeneighths, oneninth, and onetenth.

Alternative Fractions (afrc)

Replaces figures separated by a slash with an alternative form; usually stacked/ nut fractions.

Will generate a ligature lookup for these glyphs:

onehalf.afrc, onethird.afrc, twothirds.afrc, onequarter.afrc, threequarters.afrc, onefifth.afrc, twofifths.afrc, threefifths.afrc, fourfifths.afrc, onesixth.afrc, fivesixths.afrc, oneseventh.afrc, twosevenths.afrc, threesevenths.afrc, foursevenths.afrc, fivesevenths.afrc, sixsevenths.afrc, oneeighth.afrc, threeeighths.afrc, fiveeighths.afrc, seveeneighths.afrc, oneninth.afrc, twoninths.afrc,ourninths.afrc, fiveninths.afrc, seveenninths.afrc, eighnninths.afrc, onetenth.afrc, threetenths.afrc, seventenths.afrc, ninetenths.afrc, onesixteenth.afrc, threesixteenths.afrc, fivesixteenths.afrc, sevensixteenths.afrc, ninesixteenths.afrc, elevensixteenths.afrc, thirteensixteenths.afrc, fiftensixteenths.afrc, onethirtysecond.afrc, threethirtyseconds.afrc, fivethirtyseconds.afrc, seventhirtyseconds.afrc, ninethirtyseconds.afrc, eleventhirtyseconds.afrc, thirteenthirtyseconds.afrc, fifteenthirtyseconds.afrc, seventeenthirtyseconds.afrc, nineteenthirtyseconds.afrc, twentyonethirtyseconds.afrc,

twentythreethirtyseconds.afrc, twentyfivethirtyseconds.afrc,
twentyseventhirtyseconds.afrc, twentyninethirtyseconds.afrc,
thirtyonethirtyseconds.afrc, onesixtyfourth.afrc, threesixtyfourths.afrc,
fivesixtyfourths.afrc, sevensixtyfourths.afrc, ninesixtyfourths.afrc,
elevensixtyfourths.afrc, thirteensixtyfourths.afrc, fiftensixtyfourths.afrc,
seventeensixtyfourths.afrc, nineteensixtyfourths.afrc, twentyonesixtyfourths.afrc,
twentythreesixtyfourths.afrc, twentyfivesixtyfourths.afrc, twentysevensixtyfourths.afrc,
twentyninesixtyfourths.afrc, thirtyonesixtyfourths.afrc, thirtythreesixtyfourths.afrc,
thirtyfivesixtyfourths.afrc, thirtysevensixtyfourths.afrc, thirtyninesixtyfourths.afrc,
fortyonesixtyfourths.afrc, fortythreesixtyfourths.afrc, fortyfivesixtyfourths.afrc,
fortysevensixtyfourths.afrc, fortyninesixtyfourths.afrc, fiftyonesixtyfourths.afrc,
fiftythreesixtyfourths.afrc, fiftyfivesixtyfourths.afrc, fiftysevensixtyfourths.afrc,
fiftyninesixtyfourths.afrc, sixtyonesixtyfourths.afrc, sixtythreesixtyfourths.afrc.

Lining Figures (Inum)

This feature replaces selected non-lining figures with lining figures. Various characters designed to be used with figures may also have lining versions, e.g. plus.Inum, equal.Inum, multiply.Inum, divide.Inum, etc.

- If the default digits (zero-nine) are non-lining, then you could add Lining Figures named zero.Inum, etc.
- If the default digits are the preferred lining figures, there is no need to add this feature.

Oldstyle Figures (onum)

This feature replaces selected figures from the default or lining style with the Oldstyle form. Various characters designed to be used with figures may also have Oldstyle versions.

- If the default digits (zero-nine) are lining figures, then you could add Oldstyle Figures named zero.onum, etc.
- If the default digits are Oldstyle Figures, there is no need to add this feature.

Tabular Figures (tnum)

This feature replaces figure glyphs set on proportional widths with corresponding glyphs set on uniform (tabular) widths. Tabular widths are generally the default.

- If tabular figures are the default, there is no need to add this feature.

Otherwise add these:

- If the default digits are lining, but not fixed width, you could add Tabular Figures named zero.tnum, etc.
- If the default digits are lining and fixed width, you could add Oldstyle Tabular figures named zero.onum.tnum, etc.
- If the default digits are Oldstyle and fixed width, you could add lining tabular figures named zero.lnum.tnum, etc.

Proportional Figures (pnum)

This feature replaces figures set on uniform (tabular) widths with corresponding glyphs set on glyph-specific (proportional) widths.

- If proportional figures are the default, there is no need to add this feature.

Otherwise add these:

- If the default digits are fixed width, you could add accompanying Proportional Figures named zero.pnum, etc.
- If the default digits are lining figures, you could add Oldstyle Proportional Figures named zero.onum.pnum, etc.
- If the default digits are Oldstyle Figures, you could add lining proportional figures named zero.lnum.pnum, etc.

Figure it out!

To let the four features above work together when the default figures are lining tabular figures, you need to add all these:

1. zero-nine.onum (Oldstyle Tabular Figures)
2. zero-nine.pnum (Lining Proportional Figures)
3. zero-nine.onum.pnum (Oldstyle Proportional Figures)

Scientific Inferiors (sinf)

Replaces Lining or Oldstyle Figures with inferior figures (smaller glyphs that sit lower than the standard baseline, primarily for chemical or mathematical notation). May also replace lowercase characters with alphabetic inferiors.

Works with variants for glyphs with suffix .sinf (e.g. five.sinf) or inferior (sixinferior).

Slashed Zero (zero)

Replaces a default form of zero with an alternative form, which, for example, uses a diagonal slash through the counter to distinguish between 0 and O (zero and capital O).

Works with variants for glyphs with suffix .zero (e.g. zero.zero).

Localized Forms (locl)

Many scripts used to write multiple languages over wide geographical areas have developed localized variant forms of specific letters, which are used by individual literary communities. For example, a number of letters in the Bulgarian and Serbian alphabets have forms distinct from their Russian counterparts and from each other. In some cases the localized form differs only subtly from the script “norm”, in others the forms are radically distinct. This feature enables localized forms of glyphs to be substituted for default forms.

The common localized forms are:

- Catalan Punt Volat

In Catalan there are words like lletres and estrella that use a “double L”, but there are also words which contain two Ls that each belong to a separate syllable, for example col·legi. In that case a “punt volat” (flying point) is used between two Ls. This feature adds a contextual lookup which substitutes:

L periodcentered -> Lmiddledot

l periodcentered -> lmiddledot

- Dutch IJ

In the Dutch language, IJ (lange ij) is sometimes considered a ligature, or even a

letter in itself. This feature adds substitutions for IJ and ij as well as variants with acute accents (`iacute_J_acutecomb` and `iacute_j_acutecomb`).

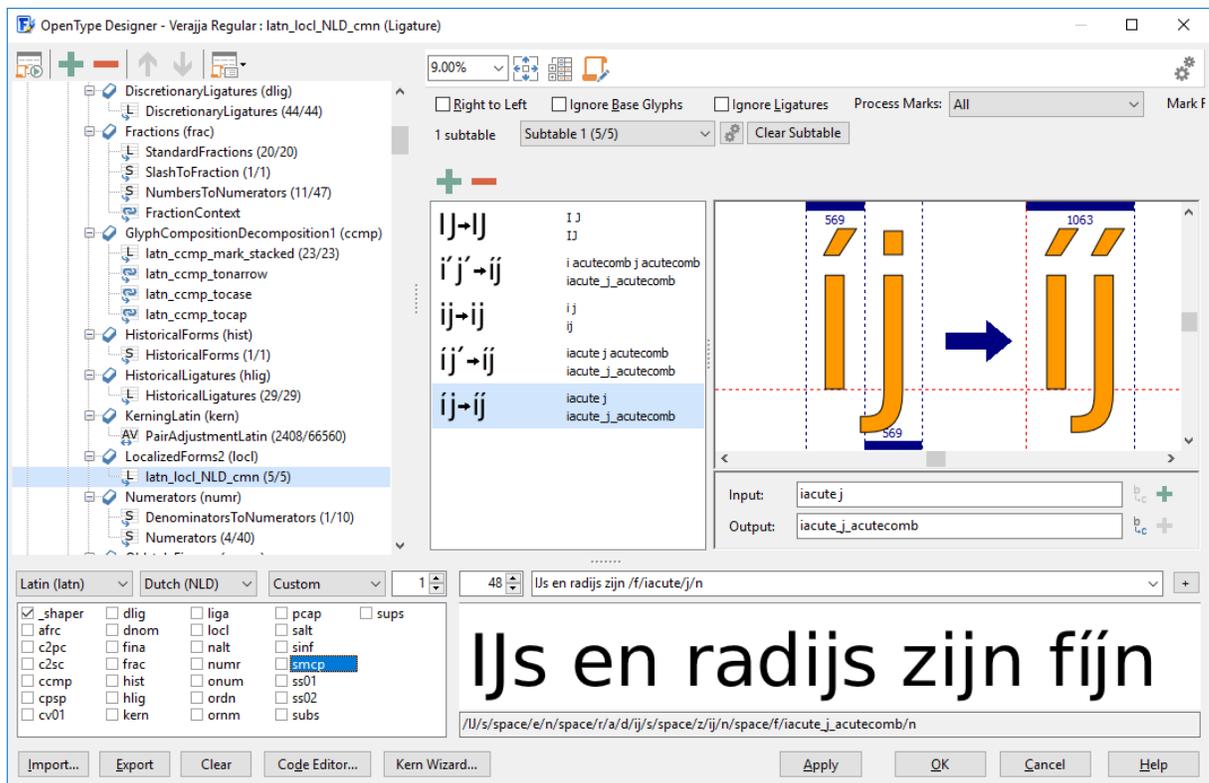
- Romanian Comma Accent

The Romanian language uses S/s and T/t with a commaaccent. Due to legacy some text and/or fonts incorrectly use S/s and T/t cedilla. This feature substitutes the commaaccent variants for both Romanian and Moldavian languages.

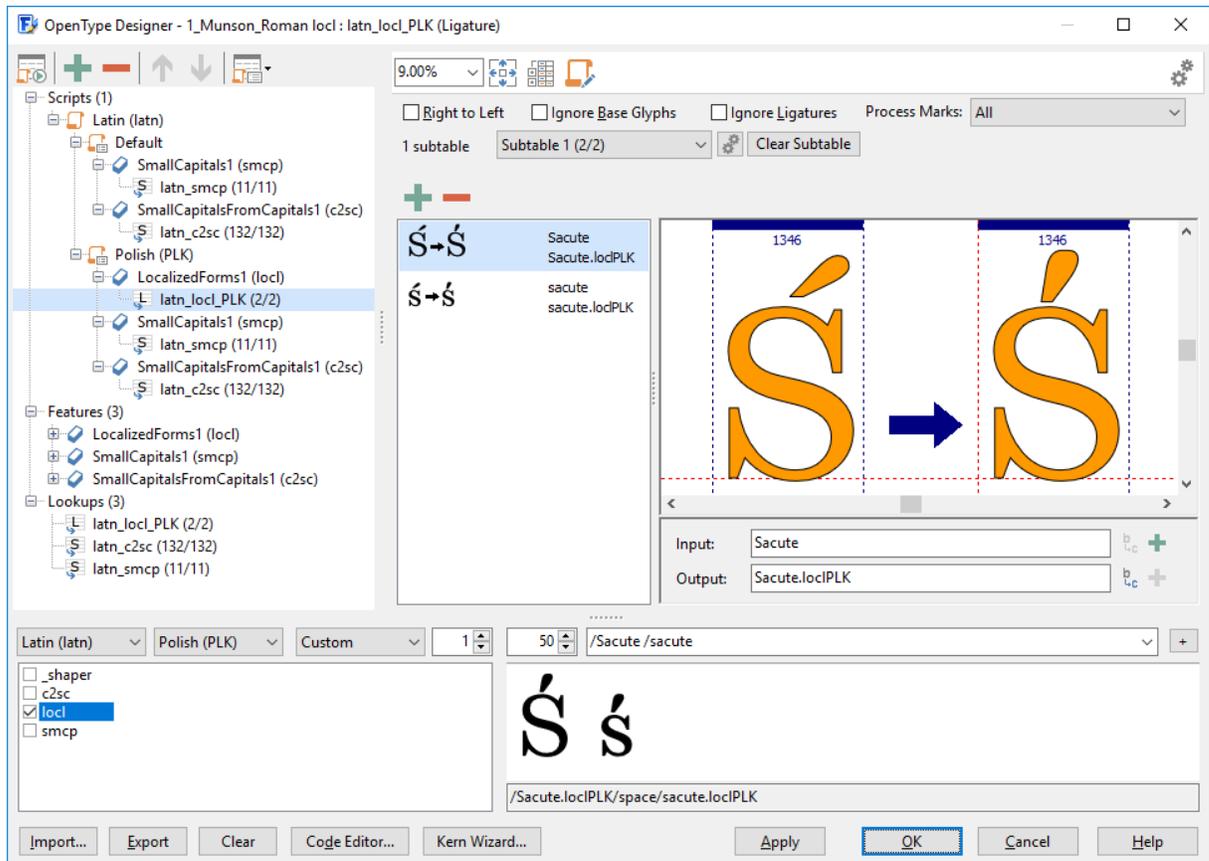
- Turkish Dotless i

Unlike most Latin based languages, Turkish have both an i with a dot and a dotless i and also the capital I and a capital I with a dot. This features prevents issues with common substitutions and mark positioning for these languages: Turkish, Azerbaijani, Crimean Tatar, Kazakh, and Tatar. This feature requires a glyph named `i.dotaccent`.

A combination of “iacute j” is common in several languages (for example the Spanish municipality Níjar), but in Dutch words the i and j belong together, so both have either a dot or acute above the base characters. However jacute is not part of Unicode, so that character does not exist. To allow for proper usage in Dutch text, fonts can make use of the `locl` feature.



Custom localized forms use glyphs with a `.locLANG` suffix to cover localized substitutions and ligatures. In this context LANG is one or more language tags separated by an underscore. For example `A.locLFRA` or `B.locLENG_ESP`. Currently only Latin based language tags, and Greek (ELL), Cyrillic (SRB, BGR), Arabic (ARA), and Hebrew (IWR) are supported. Some fonts may benefit from Polish forms, as the letters `ć`, `ń`, `ó`, `ś`, `ź` make use of the *kreska* (a diacritic used in the Polish alphabet, graphically similar to the acute accent). Add glyphs for such localized forms, naming them `Sacute.locIPLK` etc.



Tip: Use Test Web Font (CTRL + F5) to test the font on the Web Font test page. To enable localized forms in Word; select the text, apply the correct font and make sure you've activated at least standard ligatures. Finally select the correct language. To enable it in InDesign, select the language on the character panel. You can also test localized forms within the preview panel in FontCreator, just be sure you select the correct Script and Language as shown above.

Access All Alternates (aalt)

This feature makes all variations of a selected character accessible.

Works with variants for glyphs with all suffixes.

Standard Ligatures (liga)

This feature is used to map glyphs to their optional ligated form. It replaces a sequence of glyphs with a single glyph which is preferred for typographic purposes.

This feature covers these ligatures along with all glyphs with a .liga suffix:

ff, fi, fl, ,ffi, and ffl

Note: By default shaping engines should process the liga feature, but a user should be able to deactivate it. If the font has ligatures that should be formed all of the time, do define them in lig.

Historical Ligatures and Historical Forms (hlig, hist)

Replaces a sequence of glyphs with a single glyph which is preferred for typographic purposes.

This feature adds a substitution for the long form of s (longs) to hist and replaces the default (current) forms with the historical alternates to hlig for all glyphs with a .hlig suffix.

Discretionary Ligatures (dlig)

Replaces a sequence of glyphs with a single glyph covering ligatures which may be used for special effect.

This feature covers ligature glyphs with a .dlig suffix.

Note: This feature is not processed by default, but can be activated by the user.

Small Capitals (smcp, c2sc)

This option turns characters into small capitals.

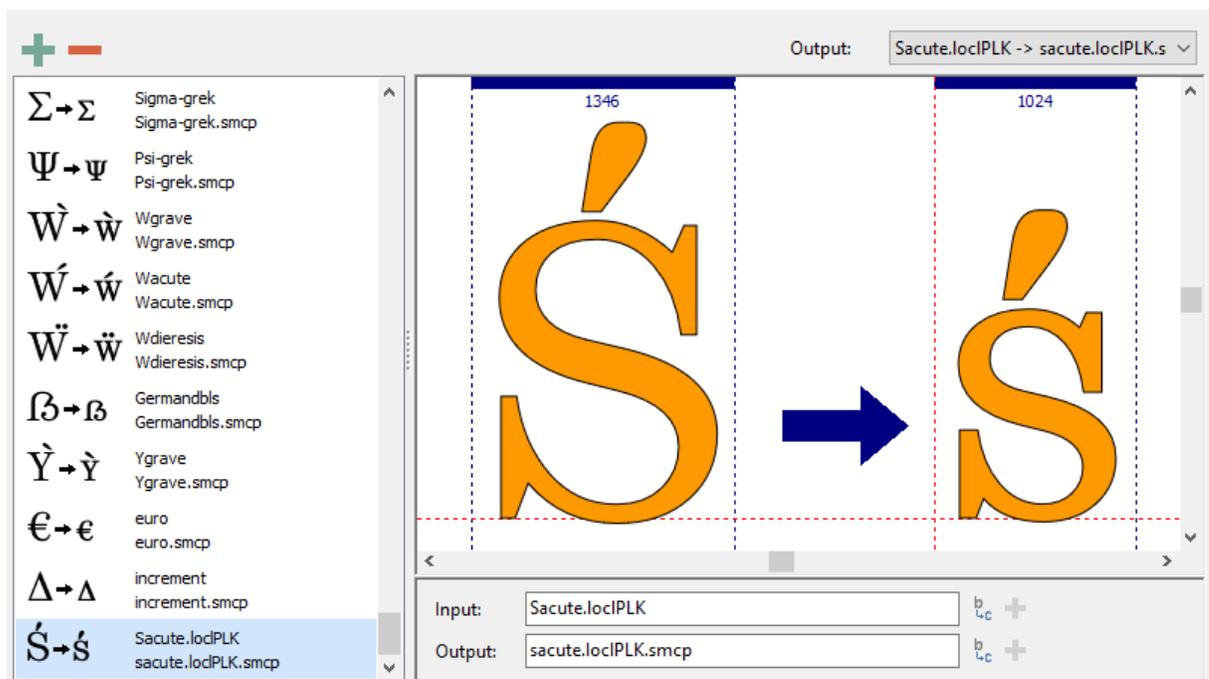
This feature covers glyphs with one of these suffixes: .smcp .c2sc .sc

We recommend to use lower case character names, for both, so if you've defined a.smcp it will be used in smcp feature (a -> a.smcp) and c2sc feature (A -> a.smcp). If

you need to differentiate between upper and lower case substitutions, then define both variants. E.g. for germandbls (German letter Eszett).

germandbls	Germandbls	germandbls.pcap	Germandbls.pcap	germandbls.smcp	Germandbls.smcp
ß	ß	ſſ	ß	ſſ	ß

You can also combine features, by adding more than one suffix. If you also include localized small caps, name those sacute.locIPLK.smcp, etc.



Petite Capitals (pcap, c2pc)

This option turns characters into petite capitals. This additional size of capital letters, shorter than the regular small capitals allow a font with a small lowercase x-height to better harmonise with lowercase text than the taller small capitals.

This feature covers glyphs with one of these suffixes: .pcap .c2pc .pc

Similar to Small Capitals, we recommend to use lower case character names, e.g. a.pcap

Subscript (subs)

Replaces a default glyph with a subscript glyph.

Works with variants for glyphs with suffix .subs (e.g. four.subs) or inferior (fourinferior).

Superscript (sups)

Replaces a default glyph with a superscript glyph.

Works with variants for glyphs with suffix .sups (e.g. four.sups) or superior (foursuperior).

Stylistic Sets (ss01-ss20)

These allow for sets of stylistic variant glyphs corresponding to portions of the character set, e.g. multiple variants for lowercase letters in a Latin font.

This feature covers glyphs with one of these suffixes: .ss01 .ss02 etc.

Note: FontCreator supports [feature parameters](#) for stylistic sets.

Stylistic Alternates (salt)

This feature links to the same substitution lookup as used with stylistic set 01 (latn_ss01).

An additional substitution lookup will be added for glyphs with a .salt suffix.

Character Variants (cv01-cv99)

A Character Variant feature should apply only to one character or a set of characters closely related in this way.

This feature covers glyphs with one of these suffixes: .cv01 .cv02 etc. If there are more than one alternate for a given character, do add an additional suffix. For example this will work in case you have three different variants for Latin capital letter Agrave and you wish to include them in cv01:

Agrave.cv01

Agrave.cv01.modern

Agrave.cv01.yourdescription

Note: FontCreator supports [feature parameters](#) for character variants.

Case-Sensitive Forms (case)

Mainly used to shift various punctuation marks up to a position that works better with all-capital sequences or sets of lining figures.

Works with variants for glyphs with .case suffix.

Titling (titl)

Replaces the default glyphs with corresponding forms designed specifically for titling. These may be all-capital and/or larger on the body, and adjusted for viewing at larger sizes.

Works with variants for glyphs with .titl suffix.

Swash (swsh)

Replaces default character glyphs with corresponding swash glyphs. Note that there may be more than one swash alternate for a given character.

This feature covers glyphs with a .swsh suffix. Similar to the approach with character variants; add an additional suffix after .swsh if you have more than one alternate for a given character.

Ornaments (ornm)

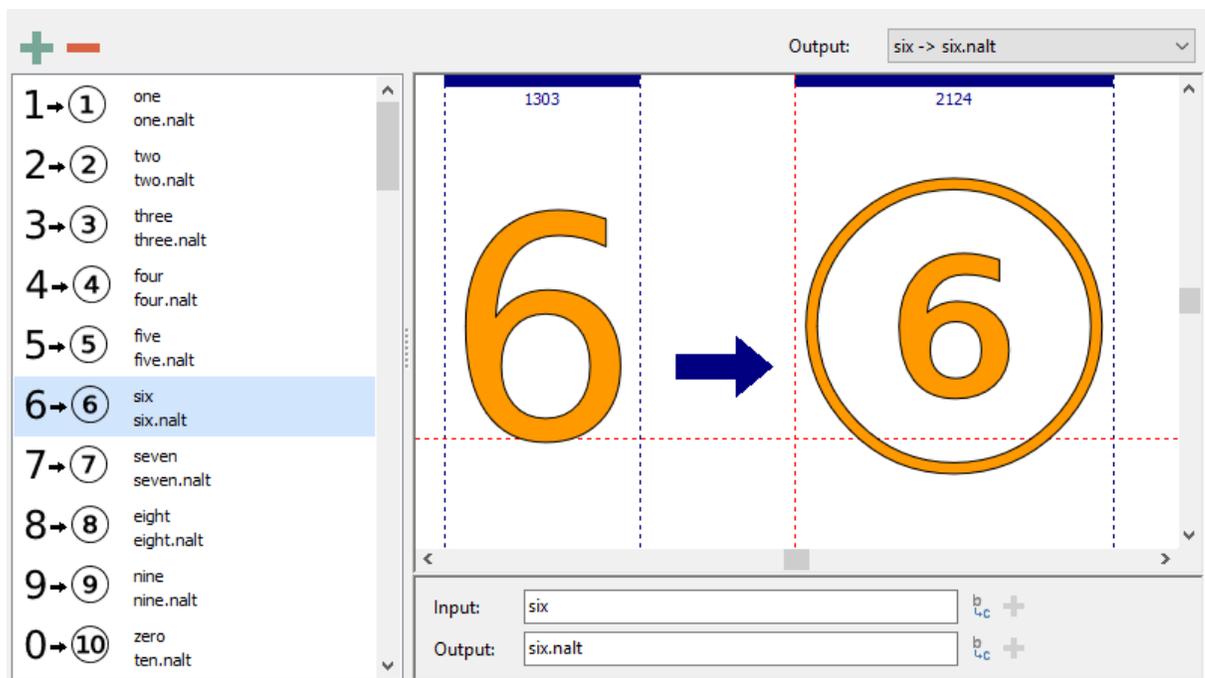
This generates a dual-function feature, which uses two input methods to give the user access to ornament glyphs (e.g. fleurons, dingbats and border elements) in the font. One method replaces the bullet character (§2022) with a selection from the full set of available ornaments; the other replaces alphanumeric characters (A-Z and a-z) with ornaments assigned to them. The first approach supports the general or browsing user; the second supports the power user.

Works with variants for glyphs with suffix `.ornm` (e.g. `a.ornm`). The first approach requires the bullet character (§2022). The second approach requires alphanumeric characters and their corresponding `.ornm` variants.

Alternate Annotation Forms (nalt)

Replaces default glyphs with various notational forms (e.g. glyphs placed in open or solid circles, squares, parentheses, diamonds or rounded boxes).

This feature covers glyphs with a `.nalt` suffix. If the font doesn't contain `.nalt` glyphs, this feature will be generated based on the circled characters as defined in Unicode block Enclosed Alphanumerics, e.g. `oneCircled`.



Ordinals (ordn)

Replaces default alphabetic glyphs with the corresponding ordinal forms for use after figures. It uses a ligature lookup to replace either “No.” or “No” with the numero character (\$2116).

The “numero” along with “N”, “o”, and “.” are required for the ligature lookup.

The extended option makes use of glyphs with a .ordn suffix, otherwise it will add substitutions for a and A to ordfeminine (\$AA) and o and O to ordmasculine (\$BA).

Initial, Medial, Terminal, and Isolated Forms (init, medi, fina, isol, med2, fin2, fin3)

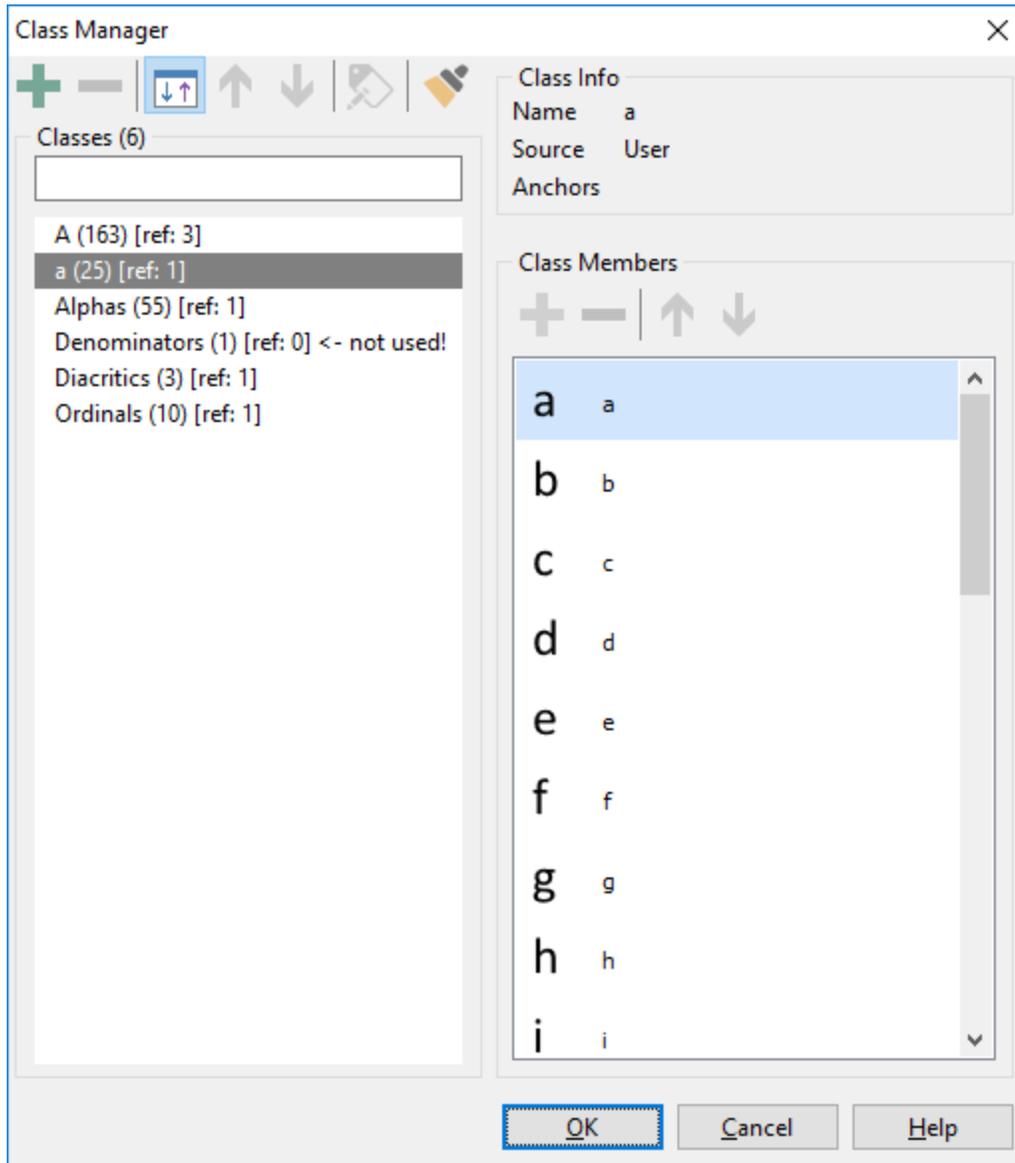
Replaces glyphs for characters that have applicable joining properties with an alternate form when occurring in an initial context.

These forms are not to be used with Latin script, but are meant for Arabic and other scripts that make use of joining properties.

Add the appropriate suffix to the glyphs you wish to cover. Some characters already come with the correct suffix to work with these options, e.g beeh-arab.init and veh-arab.fina.

Note: features med2, fin2, and fin3 are only used with Syriac script.

5.3.3 Class Manager



The class manager allows you to manage the classes used in the OpenType Designer. Here you can add, delete and rename classes and add or remove the glyphs in the class. Please note that when you delete a class, add or remove any of its members, the lookups associated with the class will also be changed. This means for example that when you add or remove a glyph to/from a class, the associated kerning pairs will also be added or removed automatically.

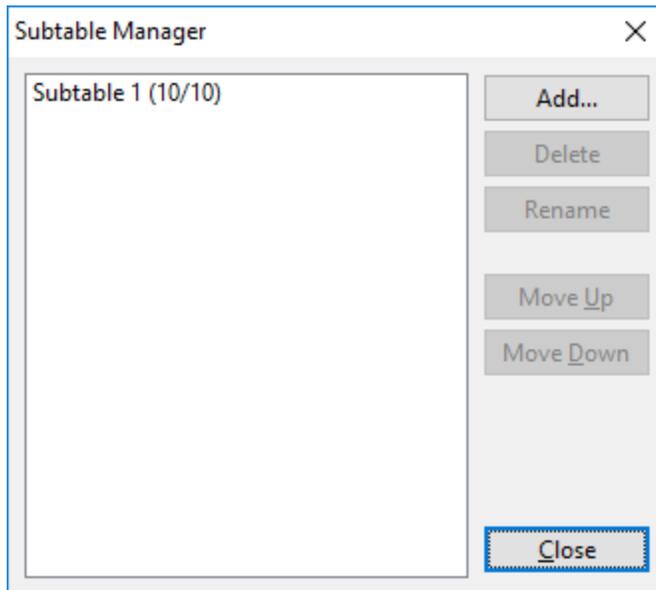
Tip: While adding class members, you can filter by anchor class. This is mostly useful when defining mark-to-mark or mark-to-base classes.

The cleanup button in the toolbar will delete all classes that are not used by any of the lookups.

Tip: Use the edit box above the classes listbox to filter the list.

Note: In general the order in which items appear in a class is not important for glyph positioning, however for glyph substitutions it is.

5.3.4 Subtable Manager



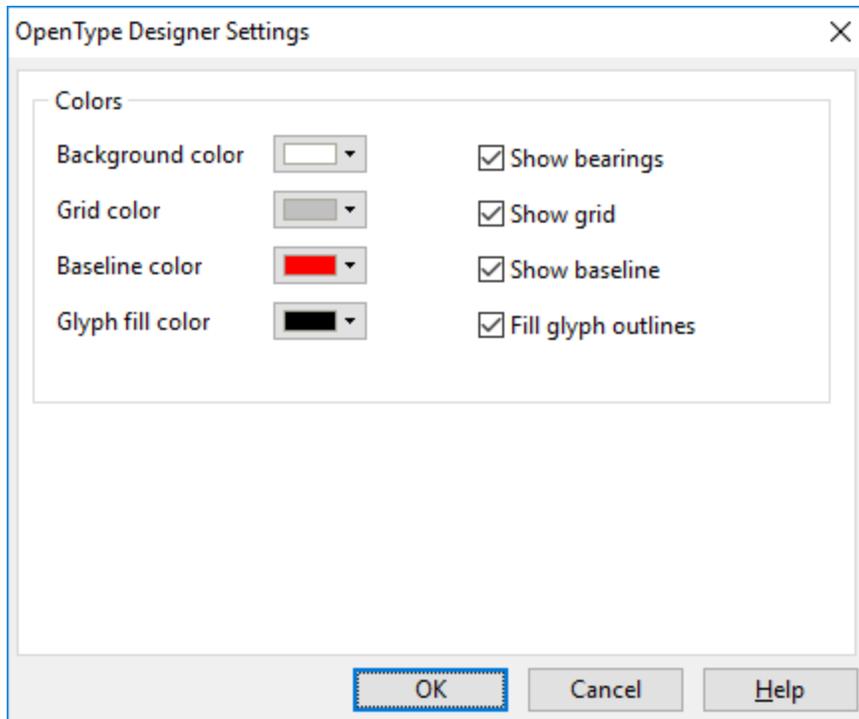
When a lookup contains a lot of entries, it is recommended (and sometimes even mandatory) to create multiple subtables to keep things organized. The Subtable Manager allows you to add, remove, rename and move subtables.

Just as with normal lookups, the order in which subtables appear controls how they are processed.

As soon as the host application has found a match in one of the subtables it will stop processing that specific lookup. This can be useful to create an "override" for kerning pairs for example. Consider a large kern lookup which contains several class-based pairs. If one or two pairs need extra modification, you can add another subtable to the lookup (ensure you've moved it up, so it is the first subtable) and add the exceptions before the actual subtable with the class-based pairs. This way the host application will find the "exception" first and will skip the "wrong" value from the large subtable.

Note: There are technical limitations to how many items can be stored in each subtable. When a font is exported, FontCreator will break up subtables into several separate subtables if necessary. We do however recommend splitting larger lookups into several subtables yourself to keep your items organized.

5.3.5 Designer Settings



The OpenType Designer Settings allow you to modify the look and feel of the designer.

Background Color

Color of the background

Grid color

Color of the grid

Baseline Color

Color of the baseline

Show Bearings

Enables/Disables the drawing of the side-bearings

Show Grid

Enables/Disables the grid

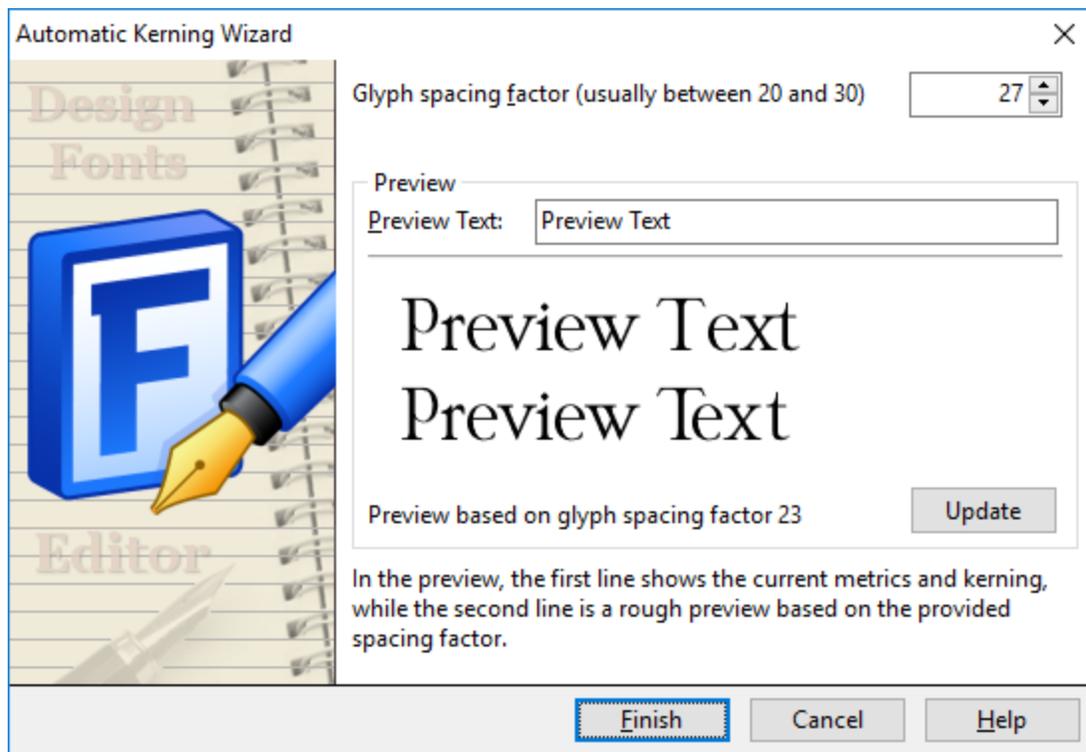
Show Baseline

Enables/Disables the drawing of the baseline

5.3.6 Autokern Settings

With **Autokerning**, you can generate new kern values for the existing kerning pairs. You can start this dialog by selecting a **Pair Adjustment** lookup in the **OpenType Designer** dialog, right-click it, and then select **Autokern**.

You can use the **Preview** area at the bottom of the **OpenType Designer** dialog to test the kerning pairs.



Glyph spacing factor allows you to define the distance between glyphs. The larger the factor the more space between glyphs, thus the larger the left and right side-bearings.

Preview is where you can define a preview text sample which will be shown in the preview area.

The **Next** button takes you to the next dialog where you can [set additional options](#).

Note: The Automatic Kerning wizard is not available in the Home Edition of FontCreator.

See also:

[Autokern new pair adjustment lookup](#)

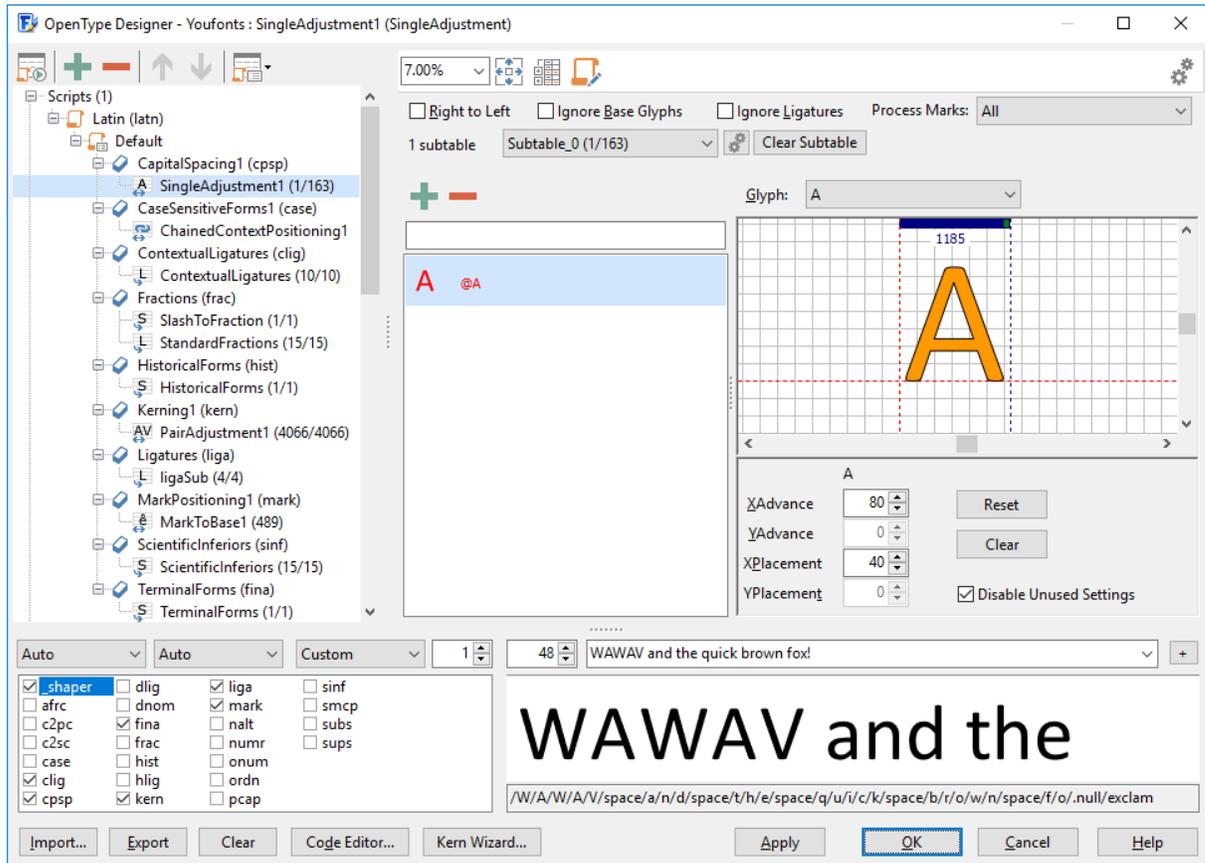
5.3.7 Substitutions

Substitution lookups allow you to supply glyph substitutes. There are several types of glyph substitution. The most common one is the ligature substitution, but which ones you want to use depends on what you want to achieve.

Within the OpenType Designer you can add specific substitution lookups to your features.

Note: Even though the OpenType specification prohibits it, FontCreator allows you to define entries in a multiple substitution lookup with no output glyphs. This is a trick to delete glyphs, and seems to be supported by Windows and web browsers.

5.3.8 Single Adjustment



Single Adjustment is commonly used to create a Capital Spacing, Superscript or Subscript feature.

On the right pane you can change the properties for each glyph separately, or when you create a class of glyphs, modify the properties of each glyph in the class simultaneously.

When you select a class, the droplist on the top of the right pane allows you to select a single member from the class, please note however that when you edit a value, it will be for all glyphs in the class at once.

To modify the values, you can either type in the values manually, or use your mouse and keyboard to drag the glyph into the right position.

- The left and right mouse buttons allow you to change the XAdvance
- By holding down the Shift key, you can change the XPlacement and YPlacement of the glyph you click on.

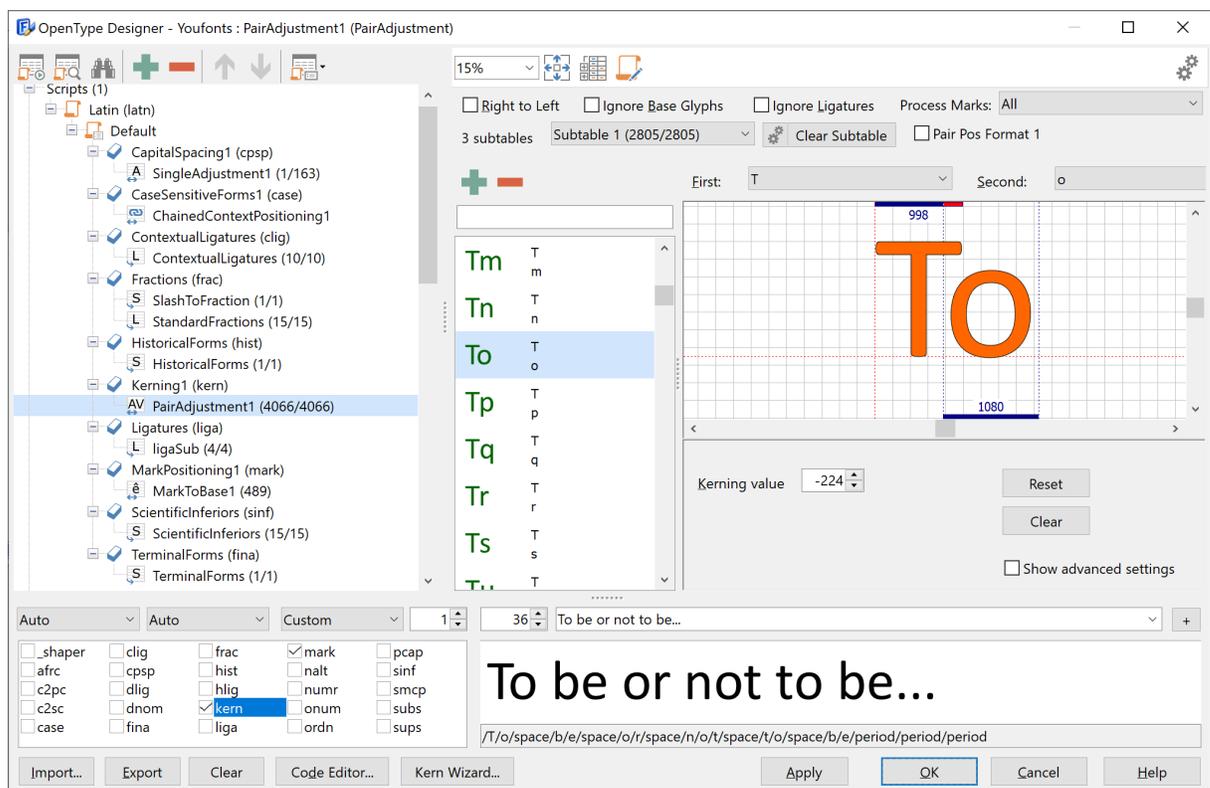
Note: For a description of the XAdvance, YAdvance, XPlacement and YPlacement fields, please see the [Pos Syntax](#)

Add...	Add a new Single Adjustment
Delete	Delete the selected Single Adjustment
Reset	Reset the values to their previous values
Clear	Set all fields to zero

5.3.9 Pair Adjustment

Pair adjustments are most commonly used to create kerning pairs.

You can use the [automatic kerning wizard](#) to generate kerning pairs for all Latin characters.



Note: Even though you can set values for the second glyph, it is recommended that you keep those values all at zero. If you do set any of those values to non-zero, the text layout engine will skip the next possible pair for adjustment, which is usually not what a font designer intends, or expects to happen.

On the right pane you can change the properties for each glyph separately, or when you create a class of glyphs, modify the properties of each glyph in the class simultaneously.

When you select a pair containing a class, the droplist on the top of the right pane allows you to select a single glyph from the class, please note however that when you edit a value, it will be for all glyphs in the class at once.

To easily identify positive and negative kern values (XAdvance for the first glyph), the listbox will show negative kerning pairs in green, positive in blue, and zero in red.

Tip: Use the edit box above the kern pair listbox to filter the list.

Tip: Select the pair adjustment lookup in the tree (left pane) and right-click to select Trim if you wish to remove kerning pairs that have a low absolute XAdvance.

To modify the values, you can either type in the values manually, or use your mouse and keyboard to drag the glyph into the right position.

- The left and right mouse buttons allow you to change the XAdvance of the glyph you click on
- While holding down the shift button, you can change the XPlacement and YPlacement of the glyph you click on.

Note: For a description of the XAdvance, YAdvance, XPlacement and YPlacement fields, please see the [Pos Syntax](#)

Add...	Add a new Pair Adjustment
Delete	Delete the selected Pair Adjustment
Reset	Reset the values to their previous values
Clear	Set all fields to zero

Glyph based pair adjustment positioning subtable (pair pos format 1)

By default all subtables without any classes defined in the pairs, are stored as glyph based.

It is sometimes convenient to build a subtable out of pairs which do contain classes, but you want it to be flattened to a glyph based subtable on export. To force this, thicken the **Pair Pos Format 1** check box.

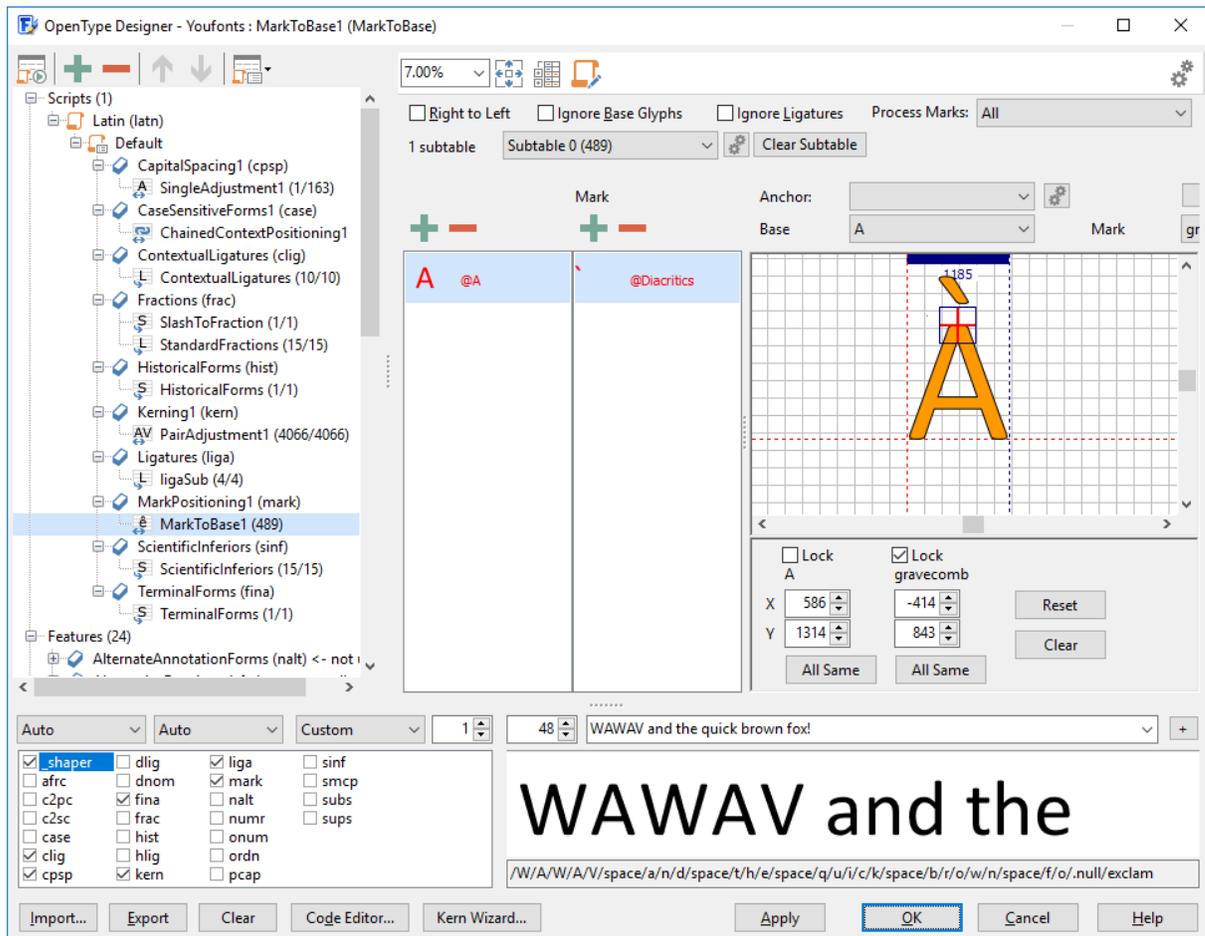
Class based pair adjustment positioning subtable (pair pos format 2)

It is important to realize class based kerning is stored in a two dimensional array; which is made out of first glyph classes and second glyph classes. If a glyph exists in a first glyph class, it should never be included in another first glyph class, as such overlapping classes are not allowed for class based kerning. Same goes for second glyph classes. Due to the nature of a two dimensional array, if a specific pair is not included in the pair list, it will still be stored in the pair positioning lookup. All glyphs that are available in the first glyph classes will cause the text processing engine to stop looking for pairs in subsequent sub lookups.

On exporting a font, all subtables containing class-based kerning will be saved in a compact class-based format. If the subtable contains overlapping classes it will be divided into as many subtables needed to meet the "no class overlap" requirement. To check whether subtables contain overlapping classes, open the table in the [OpenType Layout Feature Code Editor](#) and compile the code.

Due to limitations in the OpenType specification, it is possible a subtable is too large to be stored. In such case, the subtable will be divided.

5.3.10 Marks



Mark-to-Base, Mark-to-Ligature, and Mark-to-Mark are commonly used to add diacritical marks to base and ligature glyphs. Mark-to-Base and Mark-to-Mark are identical in usage, the only difference is that for Mark-to-Mark only mark glyphs may be used.

With Mark-to-Base, Mark-to-Ligature, Mark-to-Mark, glyphs are connected to each other via anchors. Anchors control how the mark glyph interacts with the base glyph.

Note: the number of anchor values used with Mark-to-Ligature lookups can be set through the [Glyph Properties](#).

Each base glyph has an anchor point that is defined with the left X and Y value and is indicated by the anchor icon:  Modifying these values will change the location of the mark glyph.

Each mark glyph also has an anchor point that is defined with the right X and Y value. Modifying these values will change the location where the mark glyph will be drawn in relation to the base glyph anchor.

In short:

Modify left X,Y: Change position of all marks in relation to the base glyph

Modify right X,Y: Change position of mark in relation to all base glyphs

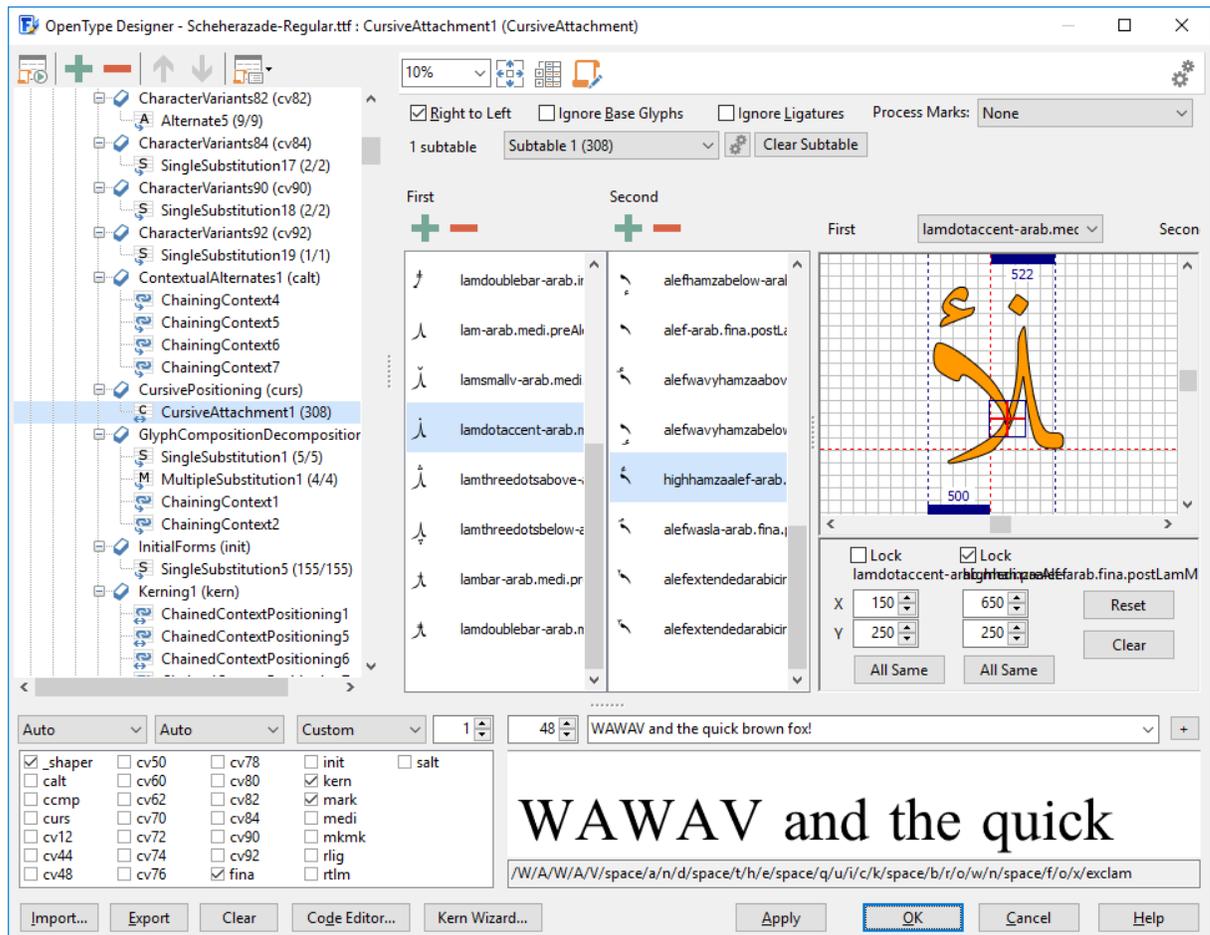
When used correctly, anchors are a very powerful tool to quickly change the position of several base/mark glyph combinations without having to modify all of your pairs. For example: if you create an anchor for uppercase glyphs and an anchor for lowercase glyphs, you could change the location of the mark of all lowercase glyphs at once, without modifying the uppercase glyphs.

< Prev	Moves to the next Base/Mark pair
Next >	Moves to the previous Base/Mark pair
Reset	Reset the values to their previous values
Clear	Set all fields to zero
All Same	This will set the same anchor value for all glyphs in the currently selected glyph class

Combining marks and signs that appear in text not in conjunction with a valid consonant base are considered invalid. In Windows, Uniscribe displays these marks using the fallback rendering mechanism, on a dotted circle. For the fallback mechanism to work properly, a font should contain a glyph for the dotted circle (U+25CC). If this glyph is missing from the font, the invalid signs will be displayed on the missing glyph shape (.notdef).

5.3.11 Cursive Attachment

A Cursive Attachment lookup is mostly used to describe cursive scripts and other glyphs that are connected with the special **Cursive** anchor class.

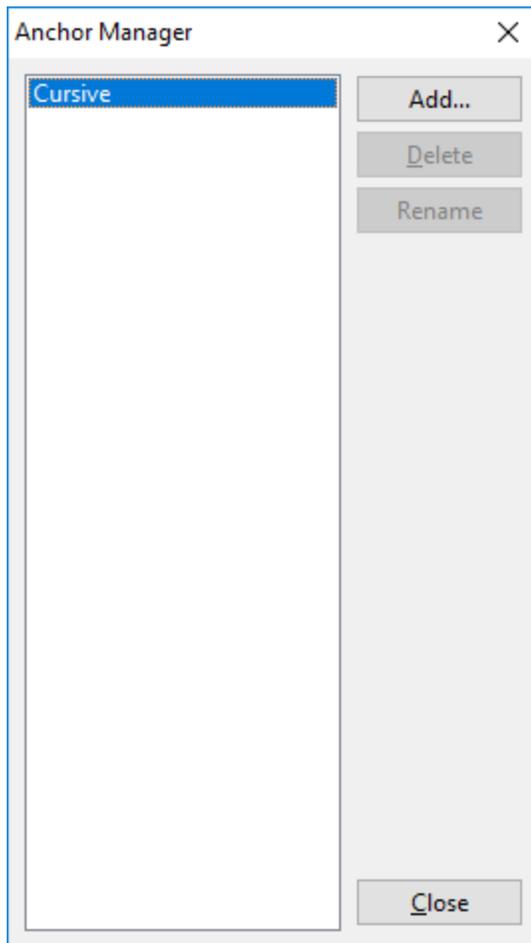


Set an exit anchor for the first glyph and an entry anchor for the second one. If placed correctly the two glyphs will be connected, as both horizontal and vertical adjustment will happen. These are mostly used with Arabic.

Right to Left

Here the Right to Left option is important. If clear, the second glyph is adjusted to align anchors with the first glyph. If set, the first glyph is adjusted to align anchors with the second glyph.

5.3.12 Anchor Manager



You can use the Anchor manager to add, remove and rename anchor classes.

Cursive is a special anchor class which is solely used with Cursive Attachment lookups.

You can add anchors to glyphs through the Anchors panel, Glyph panel, and the OpenType Designer dialog.

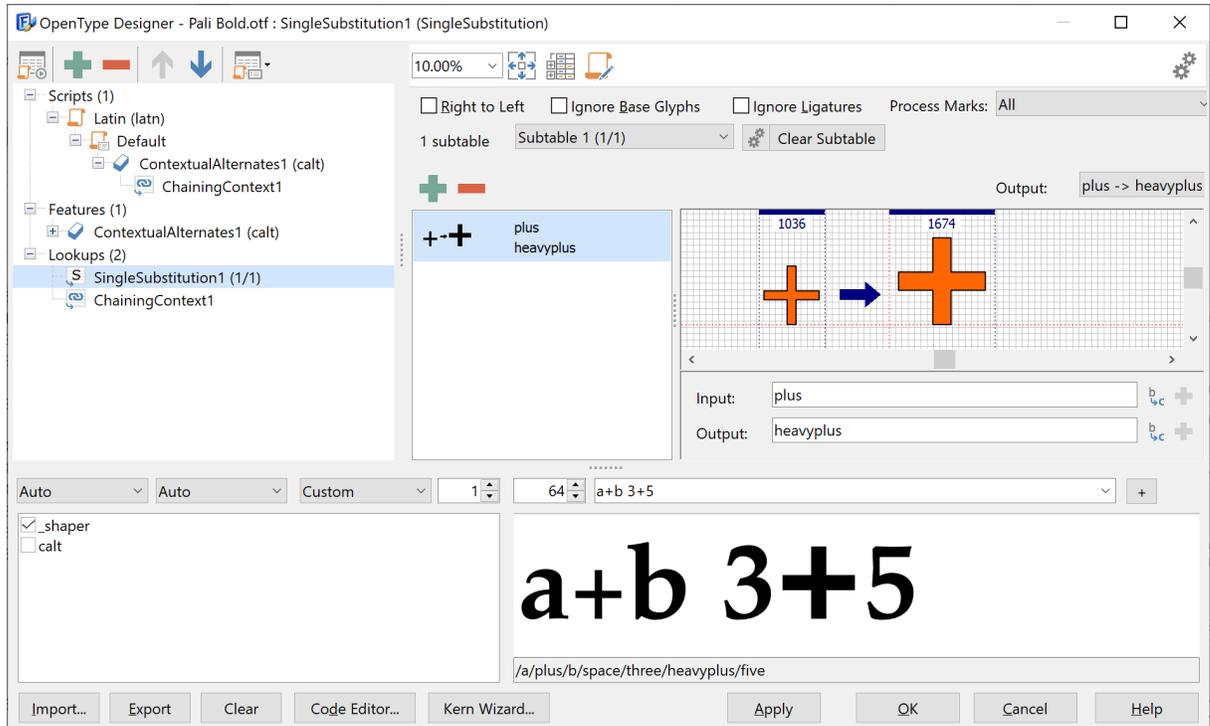
5.3.13 Chained Context

There are two Chained Context lookups, one for positioning and the other for substitutions. They will perform the substitution tables if a match is found for the combination of backtrack, input, and lookahead.

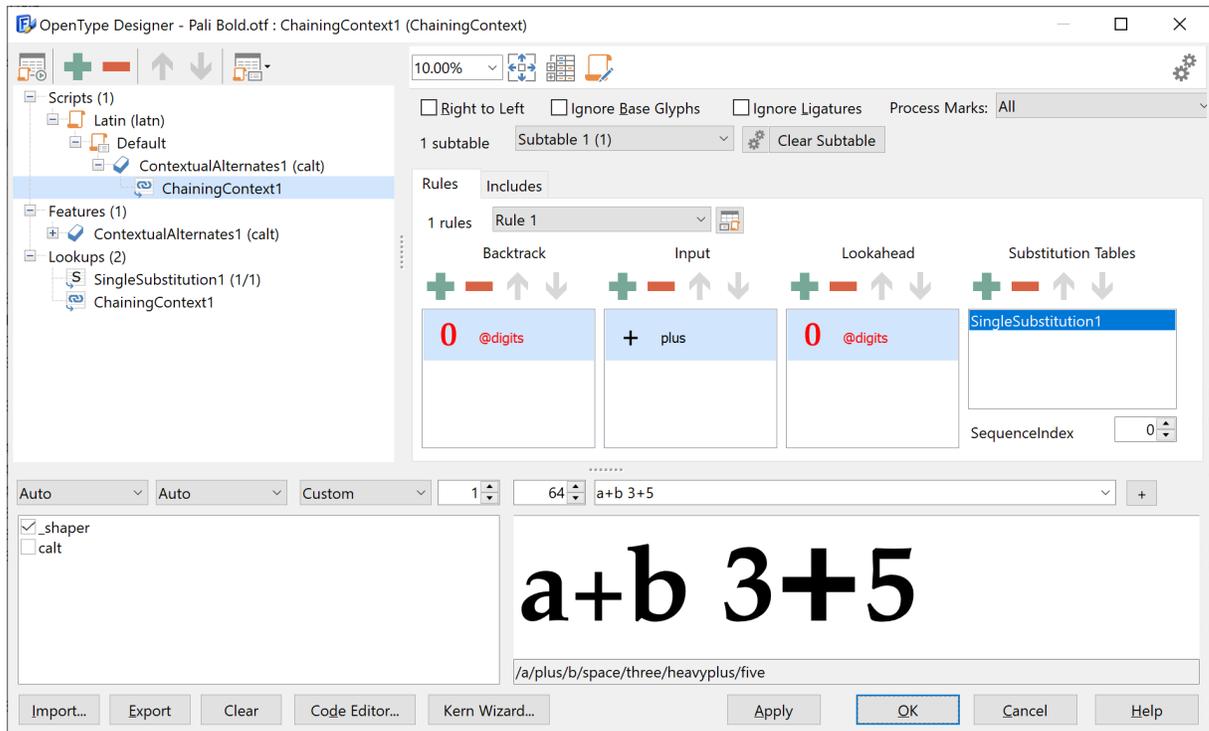
Chained Context positioning is most commonly used to change the position of certain glyphs in a specified sequence of glyphs.

Chained Context substitution is used to replace one or more glyphs in a given context.

For example, if you want to substitute the plus sign with another glyph (which shows a larger plus sign), but only if the previous and next character is a digit. Add a substitution lookup that replaces the regular plus with a heavyplus.



Next add a chaining context lookup, add digits to both backtrack and lookahead, add the plus as input, and connect the rule with the substitution lookup.



Sequence index is related to the input glyphs. By default the substitution tables will process input glyphs from the first index (which is index 0), but you can force it to apply the table from a different starting point in the input sequence by providing a non-zero sequence index.

With both chained context positioning and chained context substitution you can also make use of a special class known as class0. This class contains all glyphs except the ones used within the context. For example, if a font contains only these characters: a b c d e, then if Input uses a and d, class0 contains b c and e. The classes within the Includes section will all be treated as being not part of class0. If your rules do not make use of class0, then you can leave the Includes fields empty.

5.3.14 Feature Parameters

There are several features that support additional parameters:

- Optical size (size)
- Character Variants (cv01 - cv99)
- Stylistic Sets (ss01 - ss20)

FontCreator supports all these feature parameters. To get access to those parameters, select or add the specific feature within the OpenType Designer dialog.

Optical Size

This has been superseded by variable font technology, by means of an optical axis.

Character Variants

Feature Parameters - Character Variant - Descriptive Names and Codepoints

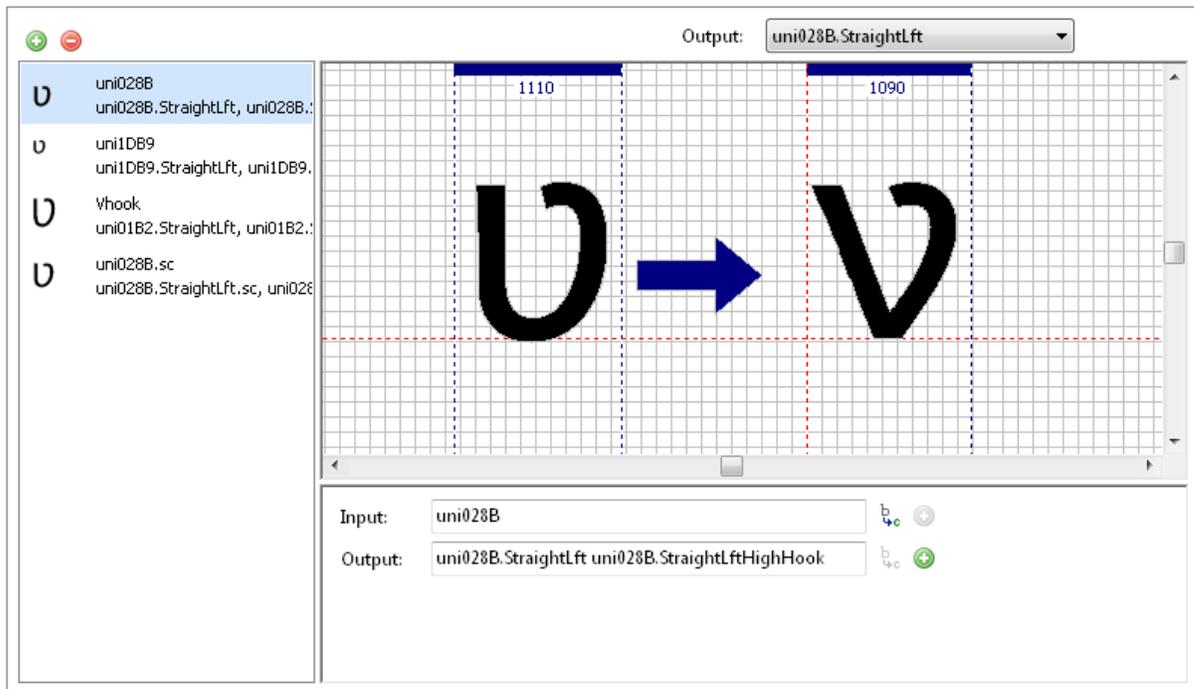
Language ID	Label	Tooltip	Sample Text
English - United States	V-hook alternates	V-hook alts	Uu°

Variant	Language ID	Content
Variant 1	English - United States	Straight with low hook
Variant 2	English - United States	Straight with high hook

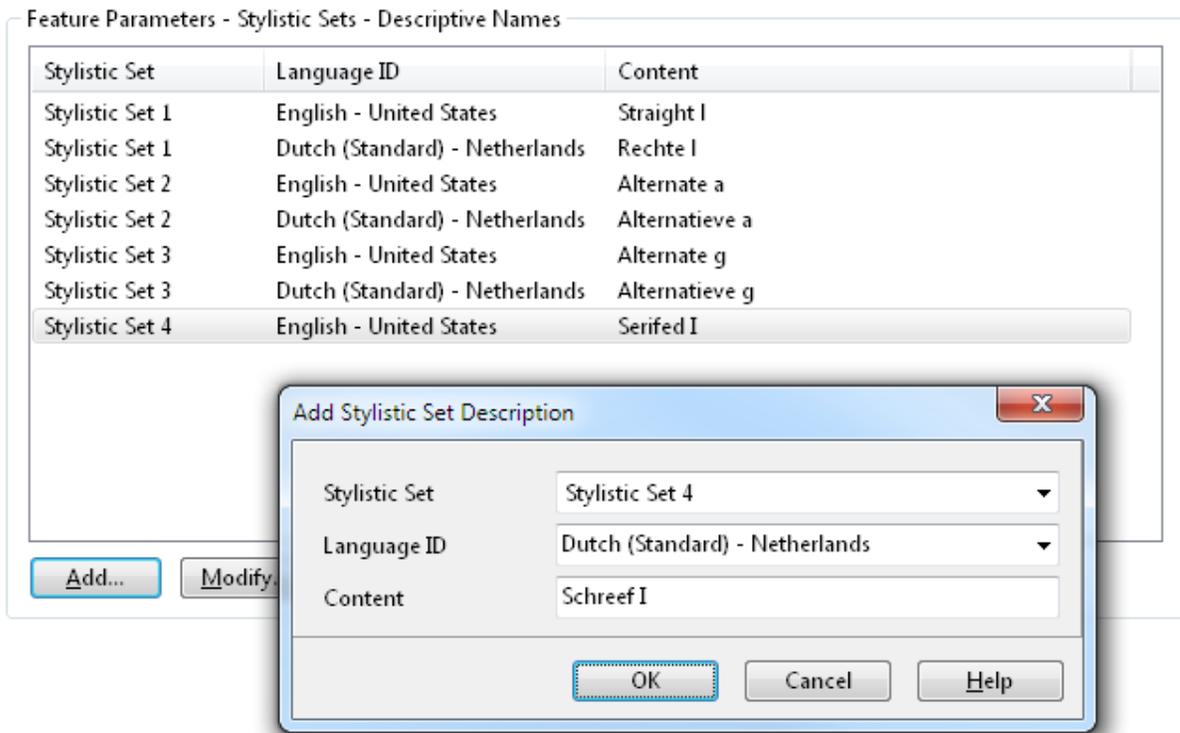
Character codepoints for which this feature provides glyph variants:

\$028B, \$1DB9

The above Character Variant feature contains the following Alternates lookup:



Stylistic Sets



As shown above, all Stylistic Sets feature parameters are grouped together.

Note: At the time of writing this manual, only a few applications make use of one or more of these parameters, but since more and more professional fonts do contain them, it is just a matter of time before we see advanced word-processing software and desktop publishing applications to support these feature parameters.

5.4 OpenType Proofing

It is important to make sure your OpenType layout features work the way they are designed. The more OpenType layout features you add to a font, the more complex and frustrating it becomes when you notice something in a line of text is not displayed correctly. It can take a lot of time to find and fix such problems. Without proper tools, it is like looking for a needle in a haystack.

If you need to debug your OpenType layout features, the proofing window comes to the rescue.

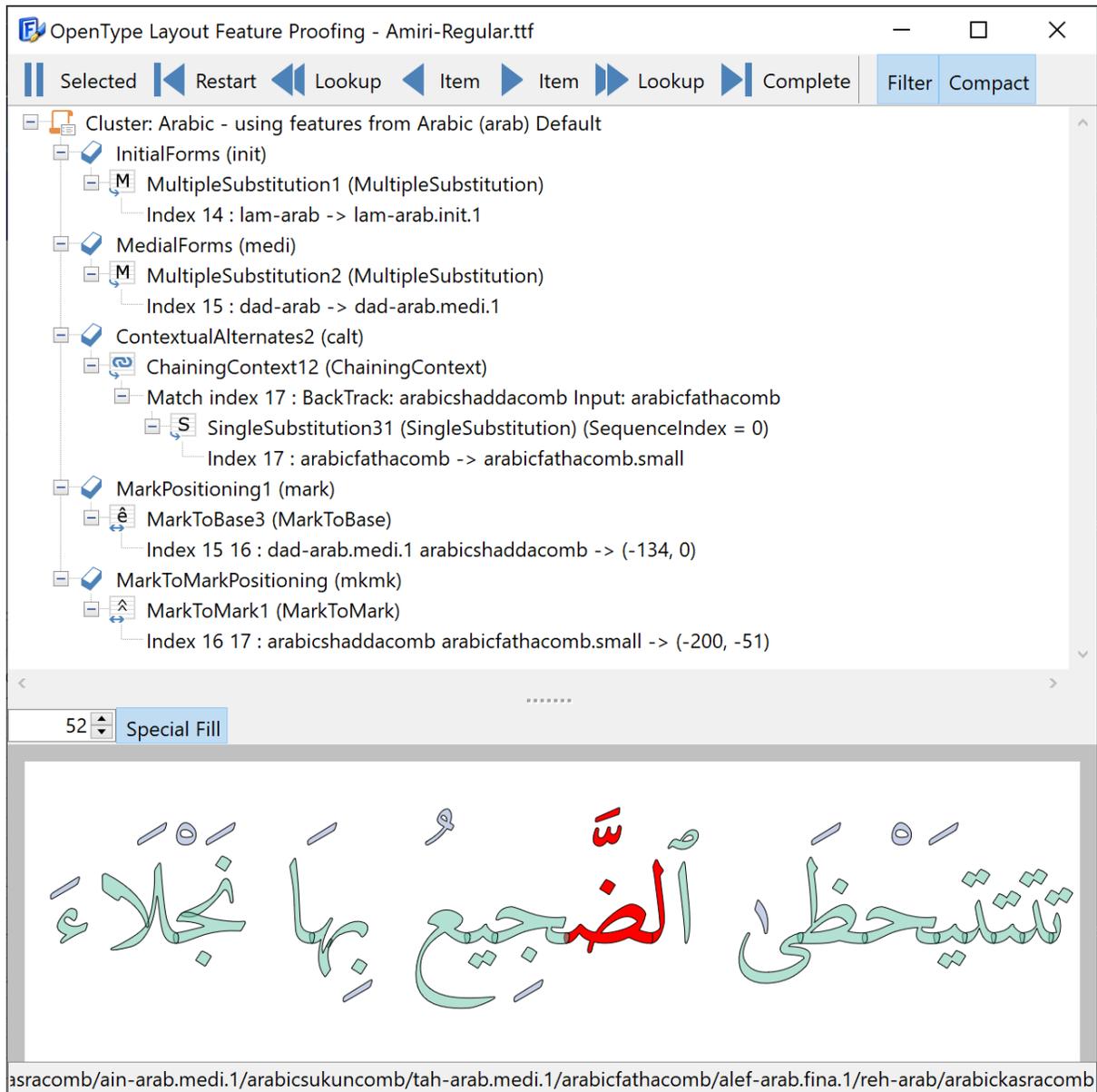
It uses the preview text and selected script, language, and features as defined in the OpenType Designer window, but the proofing window allows you to see exactly what features, lookups, and specific items have been applied to turn your original input string into the visual appearance as shown in the preview area.

The top toolbar contains several buttons that allow you to process the OpenType layout features step-by-step. The intermediate result is shown in the preview area at the bottom of the window.

Restart brings you to the beginning of the processing and Complete shows you the final result.

To zoom into a specific part of the output text, click and drag a rectangle within the Proofing preview area. This will filter only relevant OpenType layout features for the selection. You can add more to the selection by dragging another rectangle while holding down the Shift key.

Within the tree view, you can click an item, and it will be shown within the OpenType Designer window. This interactive proofing greatly speeds up the process of understanding how OpenType features work together.



In the font shown in the above screenshot, there are 33 features with 285 lookups, and totals over 24000 lines of OpenType feature code. This proofing feature allows you to only focus on those items relevant for a piece of text.

Note: the preview within FontCreator relies on the internal Shaping Engine which currently supports many but not all scripts. If you are in need of improvements, do let us know!

5.5 Script Editor

5.5.1 OpenType Layout Feature Code Editor

The **OpenType Layout Feature Code Editor** allows you to take full control over all the supported OpenType layout features in your font. If you want to automatically generate features, click the toolbar icon in the upper left corner on the [OpenType Designer](#) dialog.

Because visually adding features can be very time-consuming, the editor allows you to define and edit OpenType Layout Features for both Glyph Substitutions (GSUB) and Glyph Positioning (GPOS) through one of three supported scripting languages. FontCreator supports these script-based syntaxes:

- Adobe OpenType feature description language ([AFDKO FEA](#)), the most common known feature syntax.
- Microsoft Visual OpenType Layout Tool ([MS VOLT](#)), import and export a complete VOLT project (*.vtp) file through the OpenType Designer window.
- OpenType Layout Feature Definition (OTLFD), [our own syntax](#) based on the syntax used with OTComp.

Adobe OpenType feature description language (AFDKO FEA)

This is the preferred feature syntax. Read more about it here:

<https://adobe-type-tools.github.io/afdko/OpenTypeFeatureFileSpecification.html>

The AFDKO FEA feature syntax has several limitations, so some things that you can do within FontCreator can not be expressed in FEA syntax, namely:

- You cannot always force a specific lookup order, while shaping engines rely on the [order of lookups](#).
- FEA does not fully support class0.
- You can only add single and ligature substitutions to a single aalt feature, used by all script language pairs.
- You can only force a sub table break in pair positioning lookups.

- With large or complex lookups, you might need to manually instruct the compiler to use a special extension lookup. FontCreator does not suffer from this limitation, as it will automatically force such extension when needed.

Besides these limitations, it also supports specifying or overriding table values, which is useful on importing a UFO based font.

FontCreator fully supports the Variable FEA Syntax as proposed [here](#).

So it supports feature variation, like:

```
conditionset ConditionSet1 {  
    wght 500 800;  
} ConditionSet1;  
  
variation rvrn ConditionSet1 { # Required Variation Alternates  
    lookup SingleSubstitution35;  
} rvrn;
```

And variable positioning like:

```
markClass gravecomb <anchor (wght=400:375 wdth=75:300 wght=100:375 wght=800:375)  
(wght=400:600 wdth=75:600 wght=100:588 wght=800:620)> @top;  
and  
pos S A (wght=400:-60 wght=300:0 wght=700:0 wght=300,wdth=75:0 wdth=75:0 wght=700,wdth=75:0);
```

Microsoft Visual OpenType Layout Tool (MS VOLT)

People who use MS VOLT, can import and export such feature code, so they can work with both FontCreator and MS VOLT.

VOLT also has several limitations. For example:

- It does not support nested chained context lookups
- It has no direct way of defining sub tables
- Group (class) names are case-insensitive
- It does not support the Ignore Ligatures flag
- You can not change YAdvance in single and pair positioning.

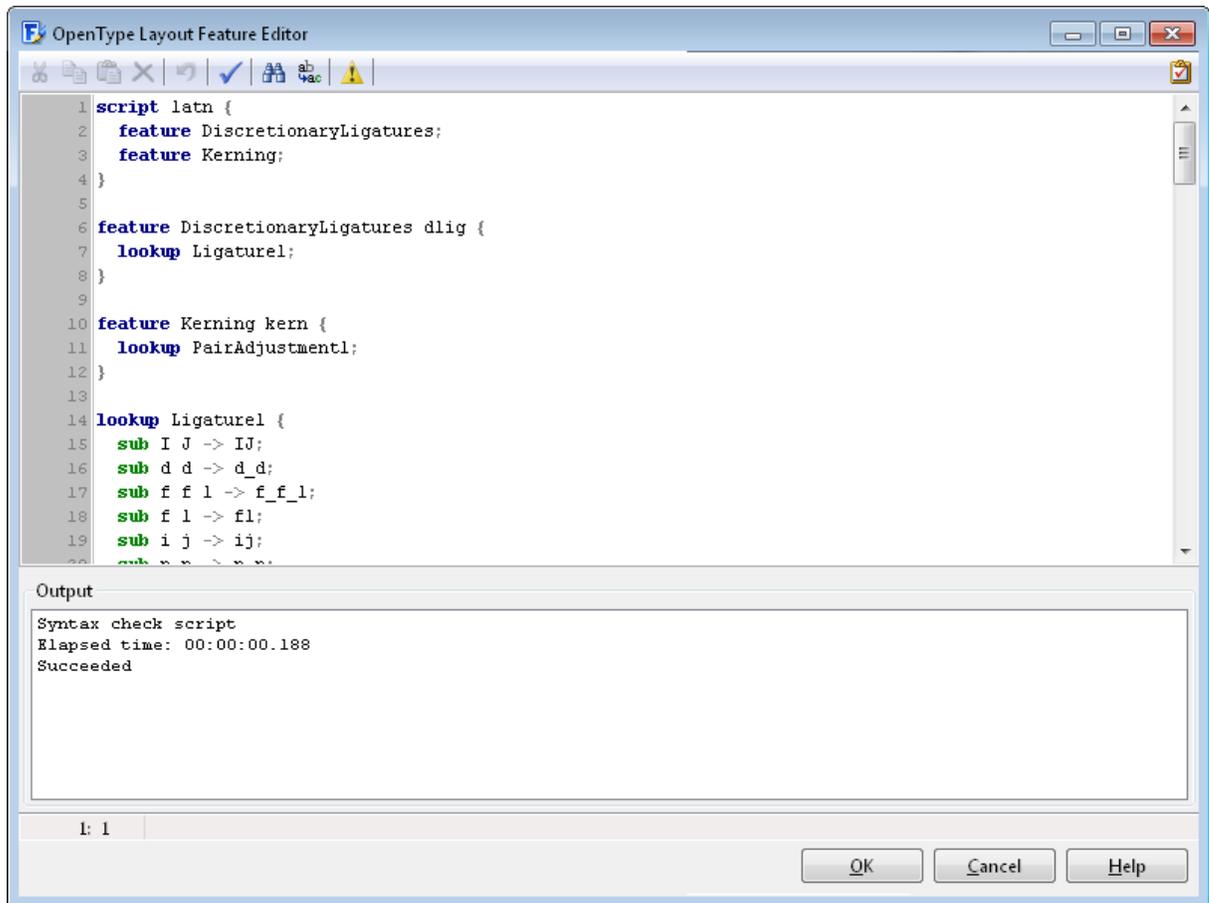
OpenType Layout Feature Definition (OTLFD)

We introduced this syntax years ago, but since most font designers are familiar with the AFDKO FEA feature syntax, we have decided to no longer maintain this syntax.

Code Editor

By default the code editor will use the Adobe fea syntax, as most font designers are familiar with it, but you can select OTLFD and VOLT from the combo box, available from the upper left corner of the OpenType Layout Feature Code Editor window.

You can update and change the script at your convenience. The find and replace feature from the toolbar might be useful for this. Shortcut Ctrl F opens the Search Text dialog, F3 finds the next occurrence of the current search text, and Shift+F3 finds the previous occurrence.



Press CTRL + Space bar to show the glyph name and class name completion helper.

To test if the syntax is correct and all **Glyph** names are valid, click the **Syntax Check button** on the toolbar or use the shortcut key F9. If there are any syntax errors or glyph names that could not be resolved, the **Output** window will list them. You can double-click on the error to quickly jump to the line where the error occurred.

If you want the compiler to ignore unknown glyphs, and allow empty classes and lookups, then click the Ignore Unknown Glyphs (warning) icon in the toolbar.

When you click the **OK** button, the code will be compiled and on success will replace all your existing scripts, features, lookups, and classes, and will merge anchors.

5.5.2 Script Syntax OTLFD

5.5.2.1 Basics

A script consist of several blocks containing declarations for scripts, features, lookups and classes. A block is started by the keyword and optional name and/or tag following a left curly bracket. A block is closed by a right curly bracket.

It is not possible to nest these blocks, with the exception of the language block which has to be nested into a script block. Names of blocks must be unique across the script and they are case-sensitive. It is possible to reference a block before it is declared. A compiler error will be generated when a feature is referenced that does not exist, and when a feature is declared but not referenced.

Basic layout of a script:

```
script <tag> {
  <feature references>
}

feature <name> <tag> {
  <lookup references>
}

lookup <name> {
  [featureflags <flags>]
  <substitution declarations>
}
```

5.5.2.2 Comments

The script-based syntax to define OpenType Layout Features supports single line comments. A single line comment ends at the end of the line. To add comments to your script, simply add a # (number sign) in front of it. Alternatively use two subsequent slashes:

```
# This is comment
// This is comment as well
```

In general, comments are not useful, as they will be lost as soon as the code is compiled into binary OpenType layout features.

5.5.2.3 Script

The script keyword is used to declare a block of features for a specific script. When the features are to be used for all scripts, the script dflt (default) should be used.

```
script <tag> {
  feature <featurename1>;
  feature <featurename2>;
  feature <featurename3>;
}
```

- **tag** is one of the script tags defined in the Microsoft list of script tags. The proposed script tags on that list are also supported. A full list of script tags can be found at: <http://www.microsoft.com/typography/otspec/scripttags.htm> 

5.5.2.4 Language

The language keyword is used to include a feature into a specific language. All features not assigned to a specific language will be added to the default language. The language keyword can only be used inside a script block. Assigning a feature to a specific language:

```
script <scripttag> {  
  <features> # available in the default language  
  language <tag> {  
    <features> # only available in this language  
  }  
}
```

Where:

- **tag** is one of the language tags defined in the Microsoft list of language tags (deprecated languages are not supported). A full list of supported languages can be found at <http://www.microsoft.com/typography/otspec/languagetags.htm> 

Don't forget to include features in the default language, as the default language will be used as fallback in case the language of text in a document isn't available in the script. Windows always uses the default language, although that might change in the future.

Note: There are fonts that have unknown language tags like "TUR " and "IPA ", these are either deprecated or common mistakes. The following table lists several of these tags and their proper replacements:

Wrong tag	Correct tag	Description
DHV	DIV	Dhivehi
IPA	IPPH	Phonetic transcription
TUR	TRK	Turkish
CHN	ZHS or ZHH	Chinese Simplified or Chinese, Hong Kong SAR

5.5.2.5 Class

The class keyword is used to declare a class of glyphs and are defined with a list of glyph names and/or class names inside a pair of brackets.

```
class @<classname> [ <classes> <glyphs> <glyph ranges> ];
```

Where:

- **classname** is a name you can create yourself. You may only use a-z, A-Z, 0-9 and _ (underscore) in your name and it must start with an @ (at sign). Names are case-sensitive, and may only be declared once.

```
class @denominators [zero.dnom-nine.dnom];
class @denominatorsandfraction [@denominators fraction];
```

Glyph classes can be used in several ways. For example:

```
lookup MyLookupTable {
  sub @lowercase -> @smallcaps;
}
```

5.5.2.6 Feature

The feature keyword is used in two ways: To reference a feature and to declare a feature. A feature is referenced from a script block to indicate that the specified feature is available in that script.

Referencing a feature:

```
script <tag> {
  feature MyLigatures; # referencing feature "MyLigatures"
}
```

Declaring a feature:

```
feature <name> <tag> {
  <feature declaration>
}
```

Where:

- **name** is a name you can create yourself. You may only use a-z, A-Z, 0-9 and "_" in your name. Names are case-sensitive, and may only be declared once.
- **tag** is one of the feature tags defined in the Microsoft list of feature tags. A full list of available tags can be found at

<http://www.microsoft.com/typography/otspec/featurelist.htm> 

5.5.2.7 Feature Params

The `params` keyword is used in two ways: To reference feature parameters and to declare them. A feature params is referenced from a feature block to indicate the specified feature params is available in that feature.

Referencing a feature params:

```
feature CharacterVariants01_1 cv01 {
    params FeatureParams_cv01;
}

feature OpticalSize1 size {
    params FeatureParams_size;
}

feature StylisticSet1_1 ss01 {
    params FeatureParams_ss01;
}
```

Declaring feature params:

There are currently three defined feature params:

Optical Size features (`size`) use the `sizeparams`:

```
params FeatureParams_size sizeparams {
    designsize 100; // the design size in decipoints
    subfamily 3; // serves as an identifier that associates fonts in a subfamily
    range 80 120; // represents the recommended usage range (start = exclusive) and
    (end = inclusive), stored in decipoints
    name 1033 "Content"; // The name for the subfamily
}
```

Stylistic Sets features (`ss01-ss20`) use the `ssetparams`:

```
params FeatureParams_ss01 ssetparams {
    name 1033 "Content for Stylistic Set 1"; // specifies text for a user-interface label
}
```

Character Variants features (`cv01-cv99`) use the `cvarparams`:

```
params FeatureParams_cv01 cvarparams {
    label {
        name 1033 "Capital-eng variants"; // specifies text for a user-interface label
    }
    tooltip {
        name 1033 "Select glyph variants for capital eng."; // specifies a tooltip text
    }
    sampletext {
        name 1033 "Content for Sample Text"; // text that illustrates the effect of the feature
    }
    variant {
        name 1033 "Content for Variant 1"; // specifies text for variant 1
        name 1033 "Content for Variant 2"; // specifies text for variant 2
        name 1033 "Content for Variant 3"; // specifies text for variant 3
    }
    characters [70 80]; // characters code-points for which glyph variants are provided
}
```

name

The name keyword requires a language id and text.

Language ID: A full list of supported Windows Language IDs can be found at <https://www.microsoft.com/typography/otspec/name.htm>. Like many values, the language id can be provided in decimal or hexadecimal notation. For example to use language English - United States, you can provide either \$409 or 1033.

Text: The text needs to be enclosed by double quote characters, unless the text uses only the following characters: a-z, A-Z, 0-9. If you want to include a double quote inside the text, you'll need to add two of them.

Note: The language ID used with the name keyword is not related to the [language](#) keyword.

5.5.2.8 Lookup

The lookup keyword is used in two ways: To reference a lookup and to declare a lookup. A lookup is referenced from a feature block to indicate the specified lookup should be used for that feature. Lookups can also be referenced from Chained context lookups.

Referencing a lookup from a feature:

```
feature MyLigatures liga {  
  lookup <name>;  
}
```

Referencing a lookup from a chained context lookup:

```
lookup MyChainedContextLookup {  
  context (@BackTrackClasses) @InputClasses (@LookAheadClasses);  
  sub 0 <lookup name>;  
}
```

Declaring a lookup:

```
lookup <name> <tag> {  
  <lookup declarations>  
}
```

Note: It is possible to set optional lookup flags (properties) via the **lookupflags** keyword.

5.5.2.9 Lookupflags

The **lookupflags** keyword is used to modify several flags (properties) of a **lookup**. The current supported flags are:

RightToLeft	This bit relates only to the correct processing of the cursive attachment lookup type (GPOS lookup type 3). When this bit is set, the last glyph in a given sequence to which the cursive attachment lookup is applied, will be positioned on the baseline.
IgnoreBaseGlyphs	Skips over base glyphs
IgnoreLigatures	Skips over ligatures
IgnoreMarks	Skips over combining marks
UseMarkFilteringSet	Indicates that the lookup table structure is followed by a MarkFilteringSet field. The layout engine skips over all mark glyphs not in the mark filtering set indicated.

Applying lookupflags to a lookup:

```
lookup MyLookupTable {  
  lookupflags <flags>;  
}
```

When you want to apply more than one flag to a lookup, simply separate them by spaces.

Note: Lookup flag names are case-sensitive.

5.5.2.10 Subtable

Sometimes lookup tables are very large and it's better to break them up into several smaller tables: for this the keyword subtable can be used. Subtables are most commonly used in kerning tables that contain a lot of kerning pairs. We recommend a subtable for every 16,000 kerning pairs.

Declaring a subtable is only possible from within a normal lookup table:

```
lookup MyKerningLookup {  
  subtable [name] {  
    <lookup declarations>  
  }  
}
```

When declaring multiple subtables, the first subtable is not required to obey the subtable syntax, but any consecutive subtable is. This means that:

```
lookup MyKerningLookup {
  <lookup declarations>

  subtable [name] {
    <lookup declarations>
  }
}
```

is valid, but

```
lookup MyKerningLookup {
  subtable [name] {
    <lookup declarations>
  }
  <lookup declarations>
}
```

is not.

The subtable name is optional, but we recommend defining one for clarity.

Note: The first declared subtable will define the lookup type and all consecutive subtables will have to be of the same type.

5.5.2.11 Sub

The **sub** keyword declares a substitute. As explained in the supported [substitution types section](#) there are several substitution types. Substitutions can only be declared in lookups. It is not possible to declare a substitution directly in a feature. Each lookup can only have one type of substitution; this means that if you want to use several substitution types in a single feature, multiple lookups have to be declared.

Declaring Single (Type 1) substitutions

```
lookup MyLookupTable {
  sub A -> a.smcp;
}
```

Declaring Multiple (Type 2) substitutions

```
lookup MyLookupTable {
  sub ffi -> f f i;
}
```

Declaring Alternate (Type 3) substitutions

```
lookup MyLookupTable {
  sub asterisk -> [asterisk asteriskmath uni2051 uni2042 uni203B uni273B];
}
```

Declaring Ligature (Type 4) substitutions

```
lookup MyLookupTable {
  sub f f i -> ffi;
}
```

Declaring Chained Context (Type 6) substitutions

```
lookup MyLookupTable {
  context (@<backtrackclasses>) @<inputclasses (@<lookaheadclasses>);
  sub 0 <substitution table>;
}
```

The order in which substitute declarations appear is also the way they are processed by applications supporting OpenType layout features. This means that in the case of ligatures:

```
lookup MyLookupTable {
  sub f i -> fi;
  sub f f i -> ffi;
}
```

is not the same as:

```
lookup MyLookupTable {
  sub f f i -> ffi;
  sub f i -> fi;
}
```

and the latter declaration will have the proper result. Why? When the sequence “f i” is encountered it will be replaced by the fi character and will no longer match the f f i sequence. In the latter example “f f i” is matched before “f i” and the result is as expected. FontCreator will automatically take care of the right order, so there is no need to worry about it.

5.5.2.12 Pos

The **pos** keyword declares a glyph positioning. As explained in the supported glyph positioning types there are several positioning types. Positioning declarations can only be declared in lookups. It is not possible to declare a positioning directly in a feature. Each lookup can only have one type of positioning; this means that if you want to use several positioning types in a single feature, multiple lookups have to be declared.

For most positioning declarations several coordinates can be defined that determine the glyph’s positioning changes. Each declaration consists of one to four coordinates.

The format of a coordinate is (including brackets):

```
<XAdvance YAdvance XPlacement YPlacement>
```

The XAdvance must always be specified, the YAdvance, XPlacement and YPlacement are optional.

XAdvance	horizontal adjustment for advance
YAdvance	vertical adjustment for advance
XPlacement	horizontal adjustment for placement
YPlacement	vertical adjustment for placement

Declaring Single adjustments (Type 1) positioning

[Single positioning](#) is used to alter the position of a single glyph or glyph class and can be used to easily create subscript and superscript like features using the same glyphs.

```
lookup MyLookupTable {
  pos A <-20 [0 0 0]>;
}
```

For classes the syntax is very similar:

```
class @class1 [A-Z]

lookup MyLookupTable {
  pos @class1 <-20>
}
```

Declaring Pair adjustments (Type 2) positioning

[Pair positioning](#) is used to alter the position of 2 glyphs or glyph classes and is mostly used to define kerning pairs.

```
lookup MyLookupTable {
  pos A B <-10 [0 0 0]> [<0 [0]>];
}
```

For classes the syntax is very similar:

```
class @class1 [A B C D E F G H I J K L M N O P Q R S T U V W X Y Z]
class @class2 [a b c d e f g h i j k l m n o p q r s t u v w x y z]

lookup MyLookupTable {
  pos @class1 @class2 <-20>
}
```

This will create 676 kerning pairs with a value of -20 with only one line of code.

It is also possible to combine a class with a single glyph:

```
class @class1 [A B C D E F G H I J K L M N O P Q R S T U V W X Y Z]

lookup MyLookupTable {
  pos @class1 hyphen <-10>
}
```

This will create 26 kerning pairs with a value of -10 with only one line of code

Force single adjustments (Type 1) positioning

Add option `PairPosFormat1`, in case you want to declare class based kerning pairs which need to be exported as flattened glyph based kerning pairs on export.

```
lookup MyLookupTable {
  option PairPosFormat1;
  pos @class1 hyphen <-10>
}
```

Declaring Mark-to-Base (Type 4) positioning

Mark-to-Base positioning is used to attach mark glyphs (diacritics) to base glyphs using anchors.

```
lookup MyLookupTable {
  mark acute 0 0;
  mark ring 0 0;
  pos A mark -10 30;
  pos B mark -10 20;

  mark esp 0 0;
  pos A mark -10 20;
}
```

Declaring Mark-to-Mark (Type 6) positioning

Mark-to-Mark positioning is used to attach mark glyphs to other mark glyphs.

The syntax is the same as for Mark-to-Base positioning, but only mark glyphs are used.

Declaring Chained Context (Type 8) positioning

This syntax is identical to the [GSUB syntax](#). The only difference is that the substitution tables are GPOS lookups instead of GSUB lookups.

5.5.2.13 Examples and Help

For example scripts and help with creating custom scripts, please read the tutorials and visit our forums at:

[FontCreator – Tutorials](#) 

<https://forum.high-logic.com/> 

Part



6 Font Tools

6.1 Tags

Tags allow you to mark glyphs so they appear with a background color in the Font panels. The same color is shown on the tab bar for glyph panels. To tag a glyph, right click on one or more selected glyphs and select one of the tags from the **Tag** submenu, use the keyboard shortcut keys (Ctrl + 1-5) or drag and drop them from the font panel onto one of the tag categories. You can view all glyphs that have a tag by selecting the **Tagged** category from the categories panel or select one of the tag subcategories to view all glyphs that have a specific tag assigned to them.

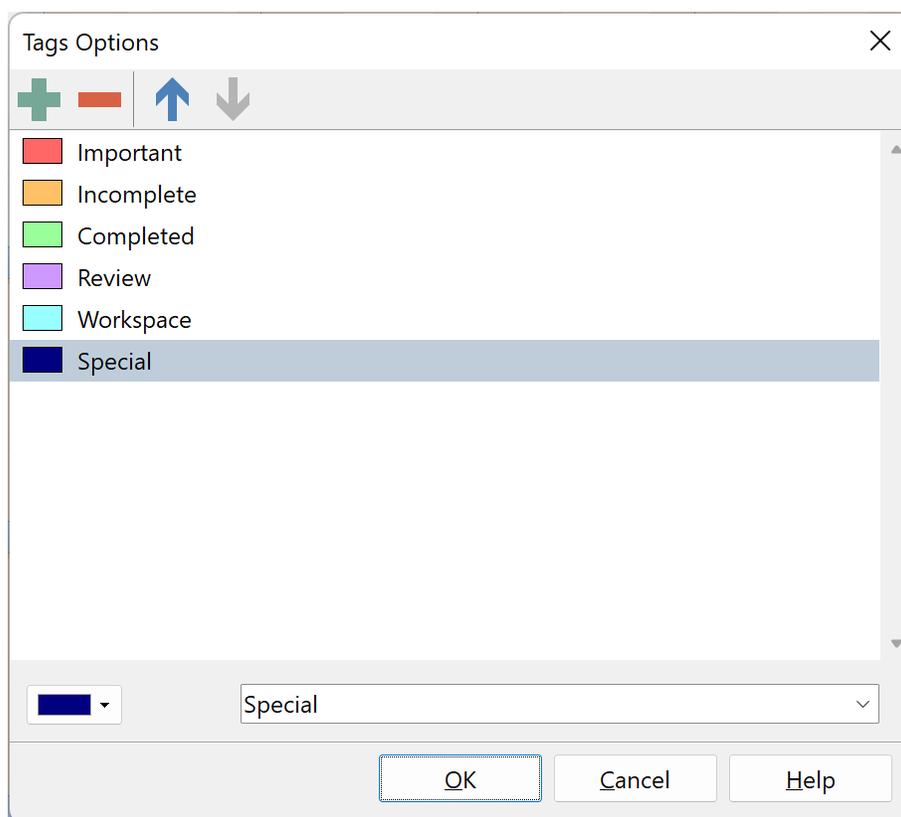
You can select glyphs with a specific tag through **Select Tagged** command from the **Edit** menu.

Toggle Tags

To untag a glyph, you can either select Tag None, or tag the glyph again.

Note: Each glyph can only have one tag, but you can also specify a tag per layer.

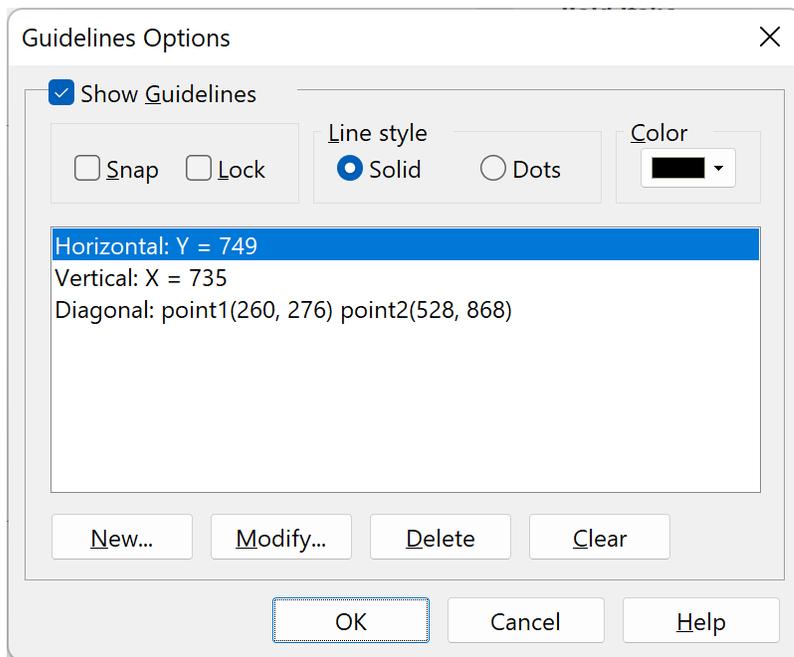
By default a new font allows you use 5 different tags, but you can easily add more through the **Tags Options** dialog. Select Tools -> Tags Options from the main menu.



6.2 Guidelines

6.2.1 Options

To open the **Guidelines Options** dialog click **Guidelines Options** on the **Tools** menu. The **Guidelines Options** window is also activated when you double-click the rulers in the **Glyph** panel.



Here you can show and hide the guidelines, and change the guidelines style and color.

The **Snap to Guidelines** function automatically places selected glyphs, contours and points along the guidelines. When you release a selection, FontCreator moves it until the edges are aligned with the nearest **horizontal** or **vertical** guidelines. It is also possible to snap points to **diagonal** guidelines. You must choose the Show Guidelines command before you can use the Snap to Guidelines function. To activate the **Snap to Guidelines** function, select the Snap check box, click the Snap to Guidelines button on the toolbar or choose Snap to Guidelines from the **View** menu. Select the **Lock** check box, or select the **Lock Guidelines** button on the toolbar, to prevent accidental movement of the guidelines.

Press the **New** button to define a new guideline. Press the **Modify** button to modify an existing guideline. To delete an existing guideline select it from the list view and click the **Delete** button. Use the **Clear** button to remove all guidelines.

If you want to add a horizontal or vertical guideline you can drag one from the top or left-hand ruler in the **Glyph** panel. Hold down the left mouse button and release it after you have moved the pointer to the desired position. To remove a guideline, simply drag it back to the ruler.

To rotate a guideline, press and hold down the Shift key before moving the guideline. Rotating a horizontal or vertical guideline will change the guideline into a diagonal guideline.

Select one or more contours (or one or more composite glyph members) and right-click and then select **Add Bounding Guidelines** to add two horizontal and two vertical guidelines that correspond to the selection bounding box.

To copy a guideline to a new position, hold down the Ctrl key as you drag the guideline.

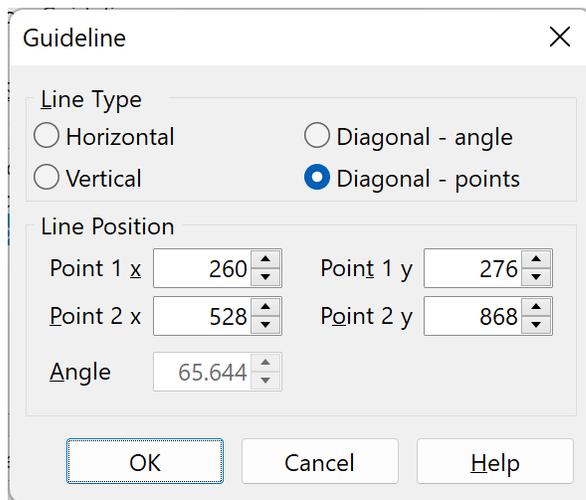
Tip: Select one point in a **Glyph** panel and press the **G** key on your keyboard to add horizontal and vertical guidelines that go through the selected point.

Tip: Select two points in a **Glyph** panel and press the **G** key on your keyboard to add a guideline that goes through the selected points.

Note: user-defined guidelines are project specific. This means you can have different user-defined guidelines for each of your projects.

6.2.2 Guideline

Use the **Guideline** dialog to create or modify guidelines.



Line Type

Choose between horizontal and vertical lines and two diagonal line types.

Line Position

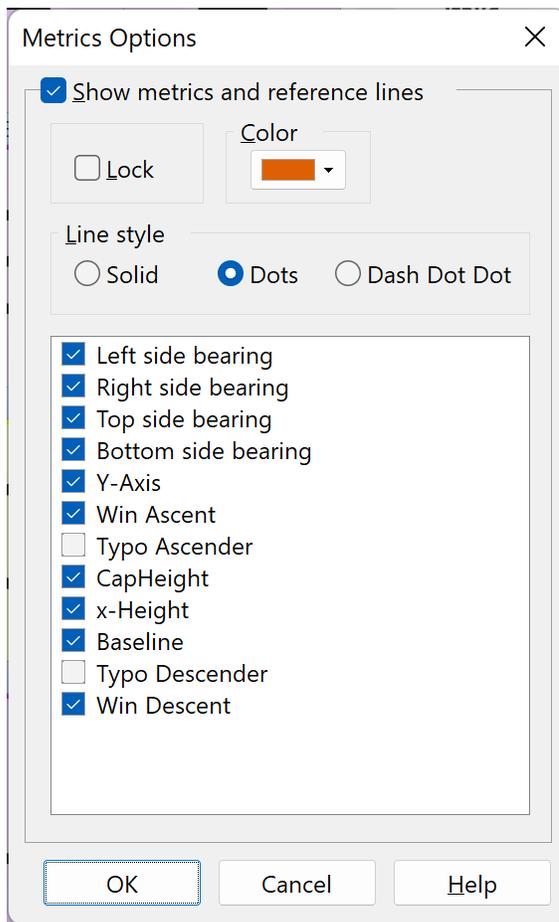
Both horizontal and vertical lines only need one value.

A diagonal line needs either one point with an angle or two points.

Tip: The **Guideline** dialog is activated when you double-click a guideline in the **Glyph** panel.

6.3 Metrics Options

To open the **Metrics Options** dialog click **Metrics Options** on the **Tools** menu. Here you can specify which metrics and reference lines should be shown in the Glyph panel.



Left side bearing The space on the left side of the glyph that is part of the advance width

Right side bearing The space on the right side of the glyph that is part of the advance width

Y-Axis The vertical line where $x=0$.

Win Ascent The ascender metric for Windows

Typo Ascender The typographic ascender for the font

CapHeight This metric specifies the distance between the baseline and the approximate height of uppercase letters

x-Height This metric specifies the distance between the baseline and the approximate height of non-ascending lowercase letters

Baseline The imaginary line upon which the letters of the font rest. This is always the X-Axis.

Typo Descender The typographic descender for the font

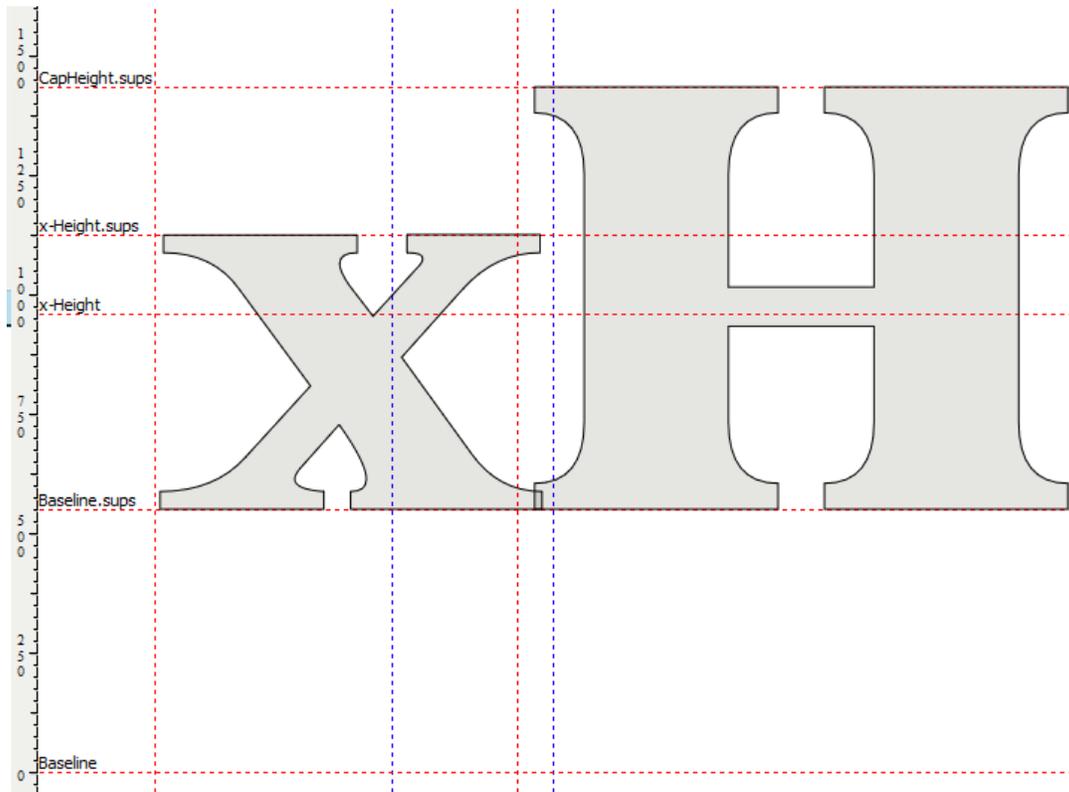
Win Descent The descender metric for Windows

You can change the position of several of these font metrics through the Metrics tab on the Font Properties dialog. From the main menu select the Font menu, click Properties, and then click the Metrics tab.

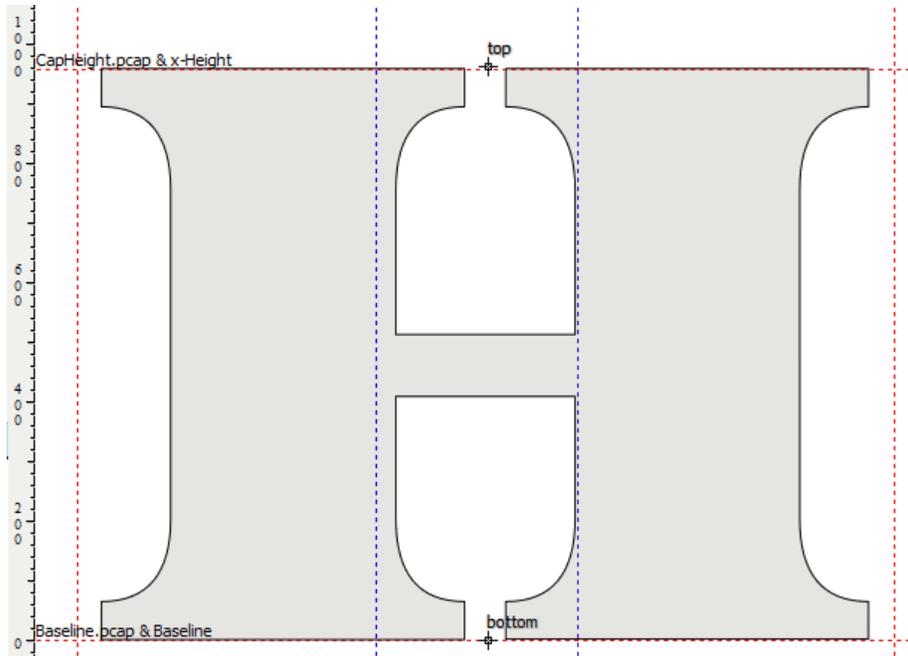
Note: Metrics options are project specific. This means you can have different metrics settings for each of your projects.

Additional metrics

More metrics will be shown if a glyph name contains a suffix. For example, if your font contains a set of superscript glyphs that have glyph names with a .sups suffix, CapHeight.sups, x-Height.sups, and Baseline.sups will also be shown. Their position is based on the top of the H.sups and the top and bottom of the x.sups.



Several Capital related suffixes (.c2sc, .smcp, .c2pc, .pcap, .pc, and .sc) will only show additional CapHeight and Baseline, based on the top and bottom of the H or h with the same suffix.

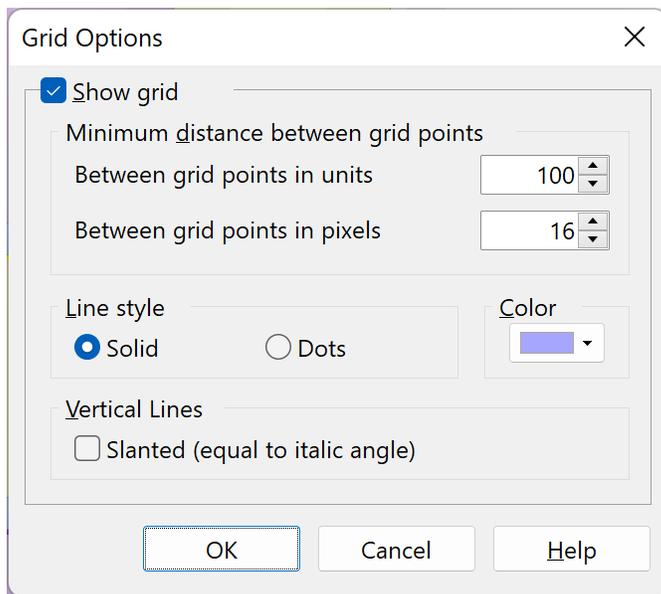


See also:

[Glyph Metrics](#)

6.4 Grid Options

You can adjust the way the grid in the **Glyph** panel is drawn in the **Grid Options** window (select **Grid Options** from the **Tools** menu).



Here you can change the minimum distance, color and style.

The **Snap to Grid** function automatically places selected glyphs, contours and points along the grid. When you release a selection, FontCreator moves it until the edges are aligned with the nearest grid lines.

You must choose the Show Grid command before you can use the Snap to Grid function. To activate the **Snap to Grid** function, click the **Snap to Grid** button on the toolbar or choose **Snap to Grid** from the **View** menu.

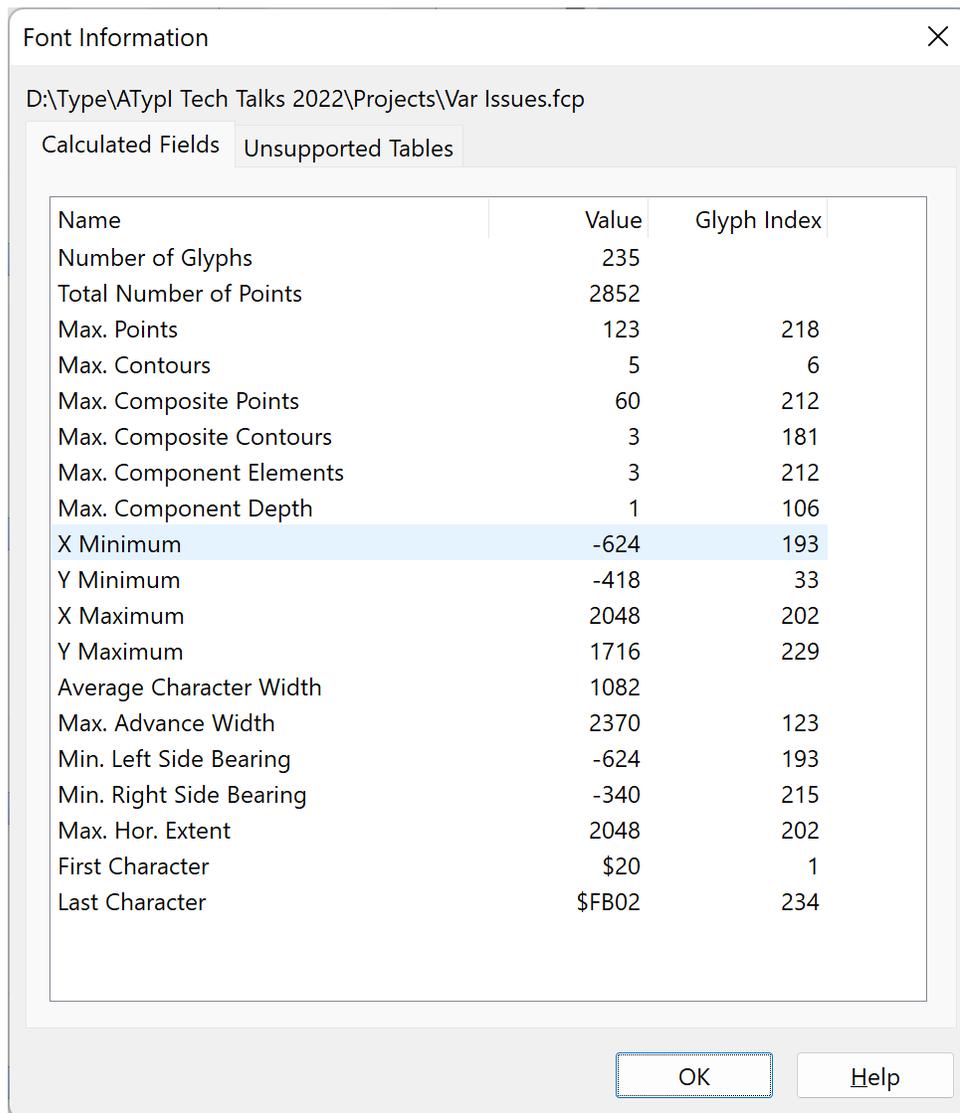
When the Snap to Grid function is active, its button on the menu and toolbar appears pressed in. Choose the command or click the button again to turn off the function.

Note: Grid options are project specific. This means you can have different grid settings for each of your projects.

6.5 Font Information

6.5.1 Calculated Fields

Information about the font is found on the Font Information dialog (from the Font menu).



These fields can't be modified directly as they are calculated and depend on other font related data, but you can double-click a row to jump to the specific glyph.

Number of Glyphs

The number of glyphs in the font.

Max. Points

Maximum points in a non-composite glyph.

Max. Contours

Maximum contours in a non-composite glyph.

Max. Composite Points

Maximum points in a composite glyph.

Max. Composite Contours

Maximum contours in a composite glyph.

Max. Component Elements

Maximum number of components referenced at “top level” for any composite glyph.

Max. Component Depth

Maximum levels of recursion; 1 for simple components.

X Minimum for all glyph bounding boxes**Y Minimum for all glyph bounding boxes****X Maximum for all glyph bounding boxes****Y Maximum for all glyph bounding boxes**

The bounding box values computed using only glyphs that have contours.

Average Character Width

Average width of all characters

Max. Advance Width

Maximum advance width value

Min. Left Side Bearing

Minimum left side bearing value

Min. Right Side Bearing

Minimum right side bearing value

Max. Hor. Extent

Maximum extent value

First Character

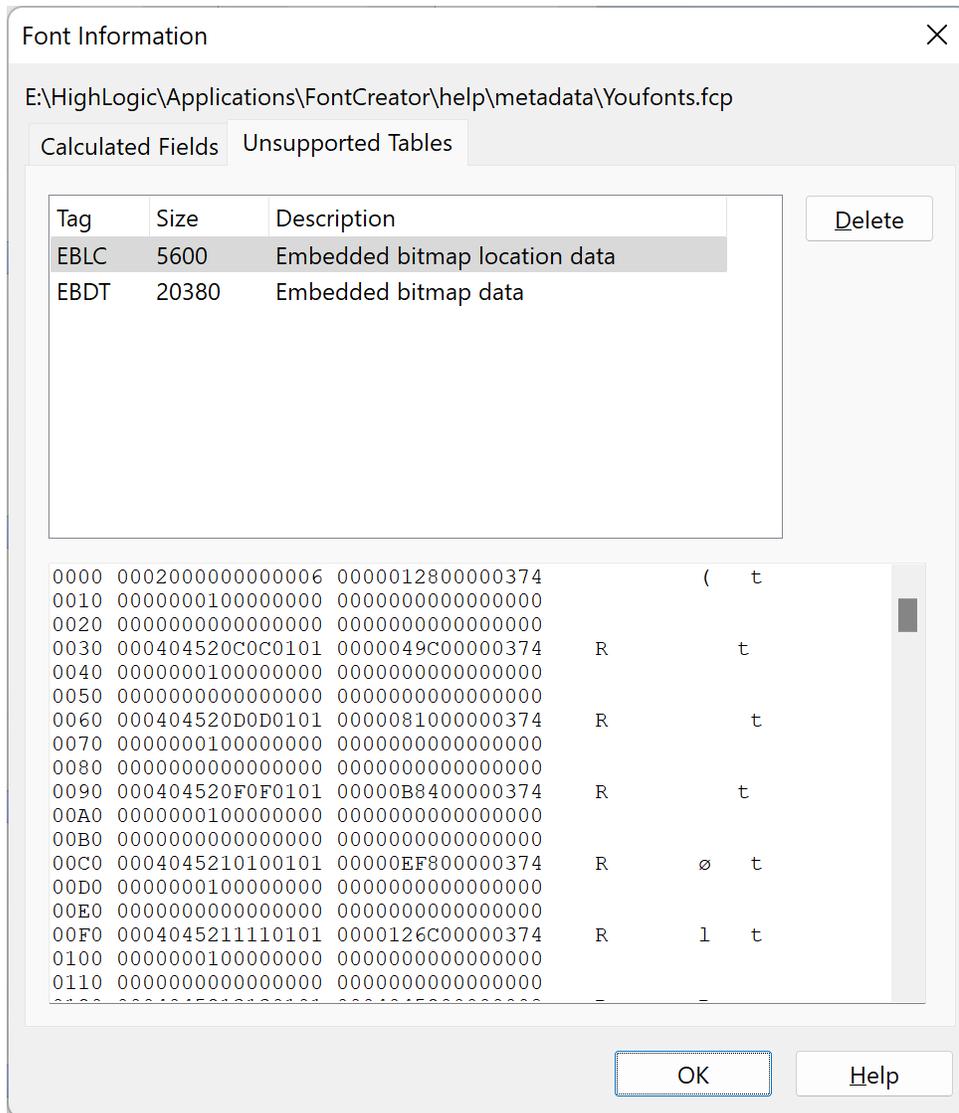
The minimum Unicode index (character code) in this font, according to the Windows Unicode BMP (UCS-2) or Windows Symbol mapping table. For most fonts supporting Win-ANSI or other character sets, this value would be 0x0020.

Last Character

The maximum Unicode index (character code) in this font, according to the Windows Unicode BMP (UCS-2) or Windows Symbol mapping table. This value depends on which character sets the font supports.

6.5.2 Unsupported Tables

Over the years the OpenType font specification has been updated with new tables and numerous tables have become obsolete. FontCreator supports all common font tables, but some tables are not supported. If you open a font that contains unsupported tables, you will see them here. On export those tables will be put back into the font without change. Sometimes this is useful, but it is also possible that change to the font cause one or more unsupported tables to become outdated.



Select a table to see the binary data in hexadecimal format.

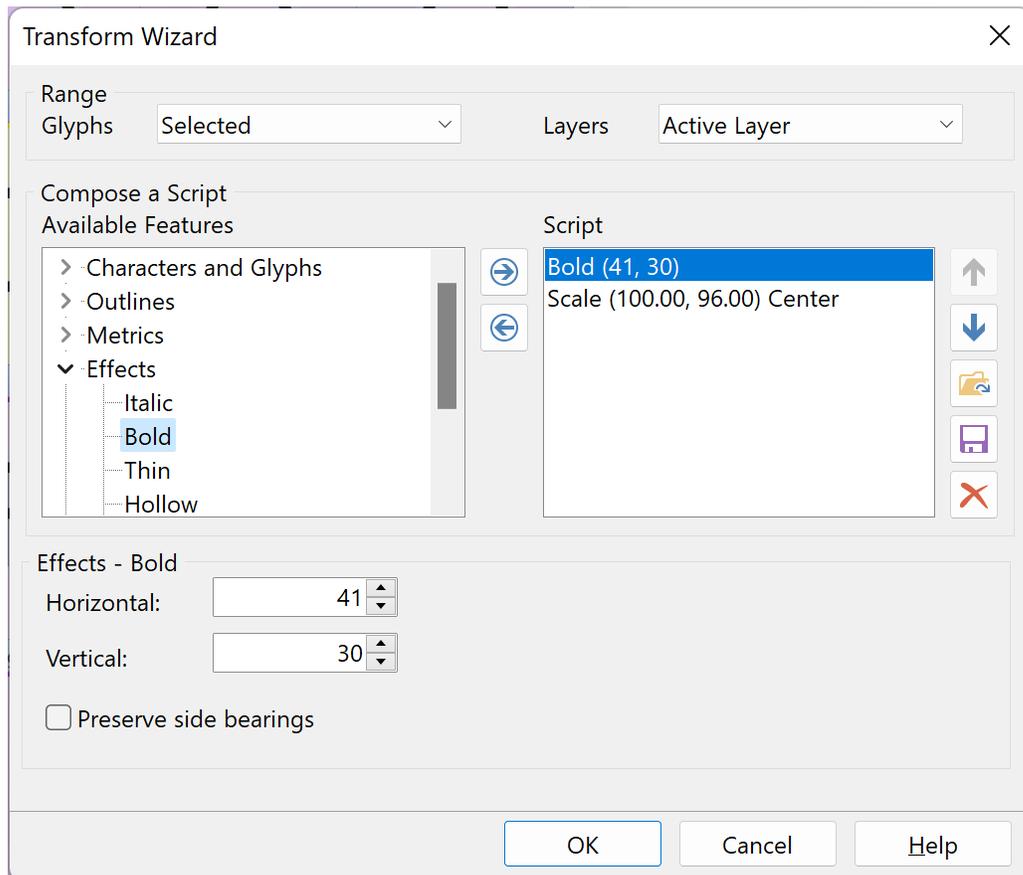
EBDT and EBLC can be considered legacy tables. font designers might decide to keep them for compatibility reasons.

If you want to delete an unsupported table, select the table on the **Unsupported Tables** page and press the **Delete** button.

Note: if your font contains unsupported tables, keep in mind that adding, deleting, or sorting glyphs might break such unsupported tables.

6.6 Glyph Transformer

The **Glyph Transformer** wizard can be selected from the **Tools** menu. Glyph transformations are scripts for changing the shape and size of glyphs, and for automating other repetitive tasks.



FontCreator comes with over 70 powerful scripts. There are scripts that allow you to change a font into an italic or bold version. Other scripts extend a font's range by adding characters for Greek Extended, Eastern Europe, Vietnamese, Ligatures, Small Capitals, and more. Several transform scripts use glyph names to insert unmapped glyphs for use with OpenType features: Petite and Small Capitals for Latin, Cyrillic, and Greek, Titling Capitals, Ordinals, Alternative Fractions, Lining, OldStyle, Proportional and Tabular Figures, Subscripts for fractions, Discretionary Ligatures, Stacking Diacritics, and Uppercase Diacritics. Each script contains descriptive comments and advice on how to use it.

These scripts can be modified to suit your needs, and you can compose custom scripts by adding commands from the list of available features on the left. Save them to use again later using the save icon, and load a saved script using the folder icon. Press the OK button to execute the currently loaded script on the current glyph in the glyph panel, on the selected glyphs, or on the entire font.

Note: Existing OpenType layout features might need to be adjusted after scaling or moving outlines and/or metrics.

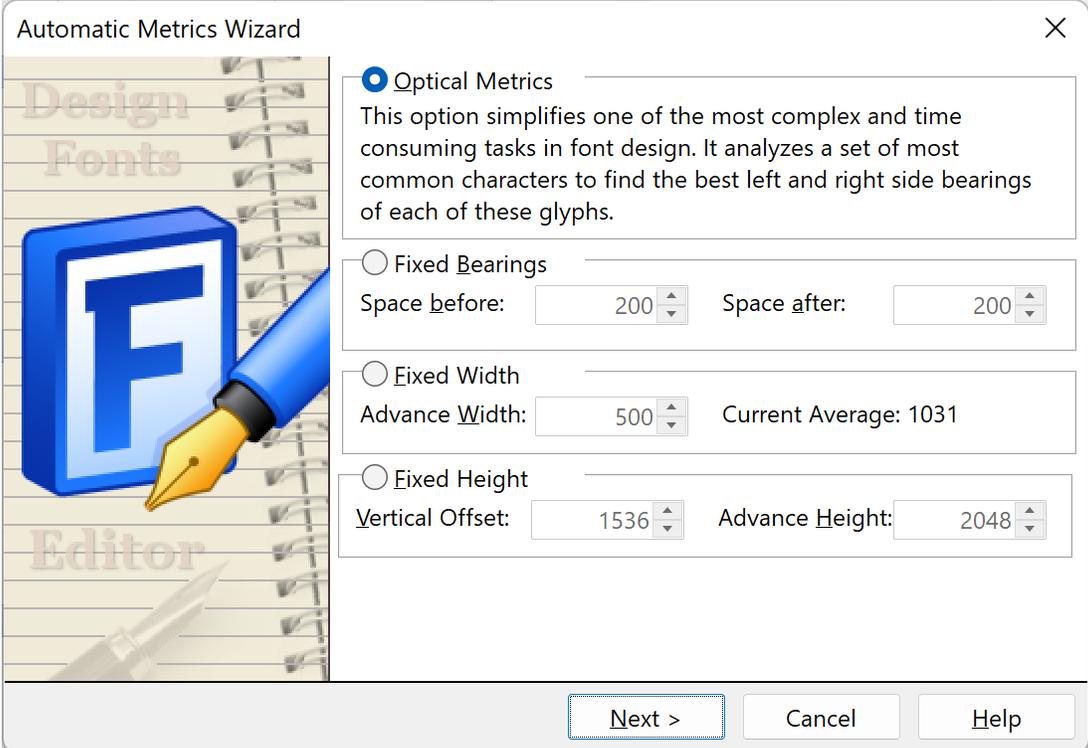
Info: If you really want to get the most out of this feature, we encourage you to read the document *Using Glyph Transformations*, available from our website:

<https://www.high-logic.com/font-editor/fontcreator/tutorials> 

Note: The Transform wizard is not available in the Home Edition of FontCreator.

6.7 AutoMetrics

With the **Automatic Metrics** wizard (select **AutoMetrics** from the **Tools** menu), you can generate the bearings for a selection of glyphs.



Automatic Metrics Wizard

Optical Metrics
This option simplifies one of the most complex and time consuming tasks in font design. It analyzes a set of most common characters to find the best left and right side bearings of each of these glyphs.

Fixed Bearings
Space before: 200 Space after: 200

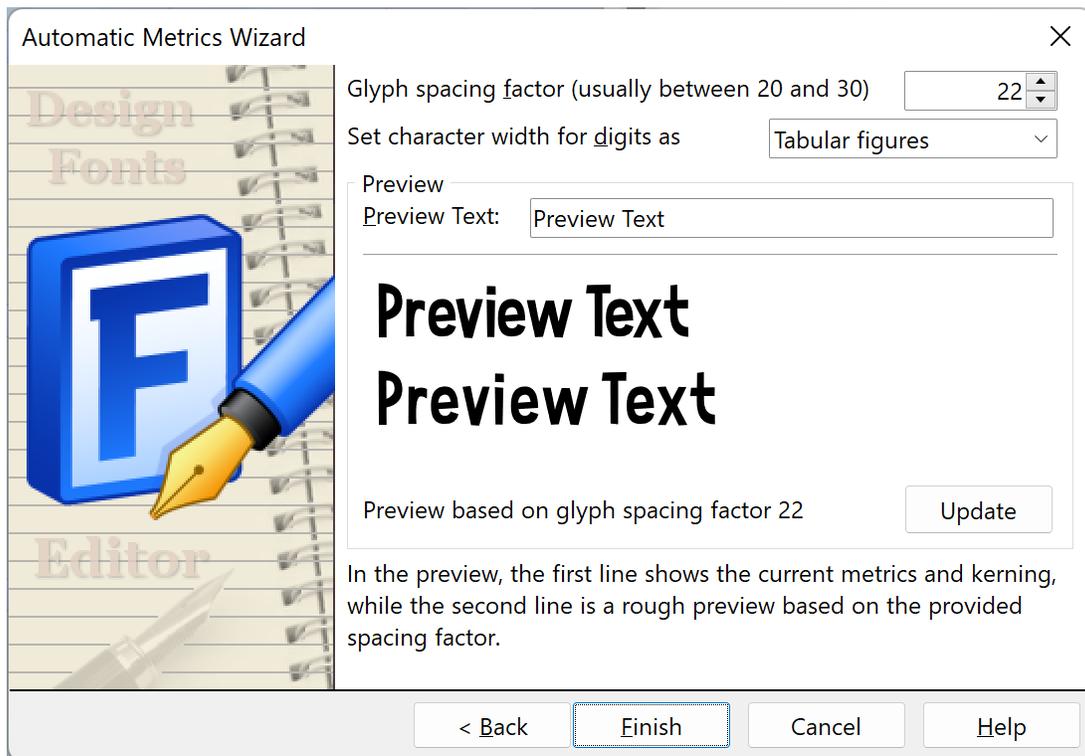
Fixed Width
Advance Width: 500 Current Average: 1031

Fixed Height
Vertical Offset: 1536 Advance Height: 2048

Next > Cancel Help

Optical

In Optical mode all Latin characters are analyzed to find the best optical space before and after each character. Please note that this process can take from several minutes up to several hours.



Glyph spacing factor allows you to define the distance between glyphs. The larger the factor the more space between glyphs, thus the larger the left and right side-bearings.

Character width for digits allows you to choose how to calculate the advance width for digits:

- **Tabular:** It is common for digits to all have the same advance width, as it allows numerals to align vertically in tables and financial statements.
- **Proportional:** All digits have their own advance width.

Preview is where you can define a preview text sample which will be shown in the preview area.

Note: This feature is not available in the Home and Standard editions of FontCreator.

Note: Add specific OpenType Layout Features if you want to include both Tabular and Proportional Figures.

Fixed Bearings

In Fixed Bearings mode the white spaces before and after the selected glyphs will be modified.

Fixed Width

In Fixed Width mode the advance width will be modified so all selected glyphs have the exact same width. This is especially useful for converting a proportional font into a monospaced font.

Note: Fixed Width might also be useful for the digits (characters 0 to 9), it is common for all of them to have the same advance width..

Fixed Height

In general vertical metrics are only useful with fonts that are used for vertical writing. So we recommend to only include them in CJK (Chinese, Japanese, and Korean) fonts. To make use of this option, first ensure you have enabled "Include metrics for vertical writing mode" on the Font Properties panel, Masters tab.

Press the Finish button to return to the **Font** panel to check the result.

6.8 AutoKern

6.8.1 Setup

With the **Automatic Kerning** wizard, you can generate kerning pairs for all Latin characters. The pairs will be added to a new pair positioning lookup which is added to the kern feature which is available for Latin script with default language. You can start this wizard by selecting AutoKern from the Tools menu, alternatively you can open this wizard from the OpenType Designer dialog.

Kerning is the reducing/increasing of the space allocated between two glyphs to make them fit more comfortably. Sometimes you want the side-bearings to be

different in special situations. When you want to change the distance between two characters you could use kerning pairs. For example the A and the V could be closer (AV) together than TV.

First, it is important to ensure that the left and right side-bearings are all set correctly for the individual glyphs. This can be done manually through the [Glyph Properties panel](#), or automatically through the [Optical Metrics feature](#).

Most but not all Operating Systems and applications support kerning. If they don't support kerning they simply ignore the kerning pairs. Many sophisticated word processors, most desktop publishing (DTP) applications, and all modern web browsers have kerning support.

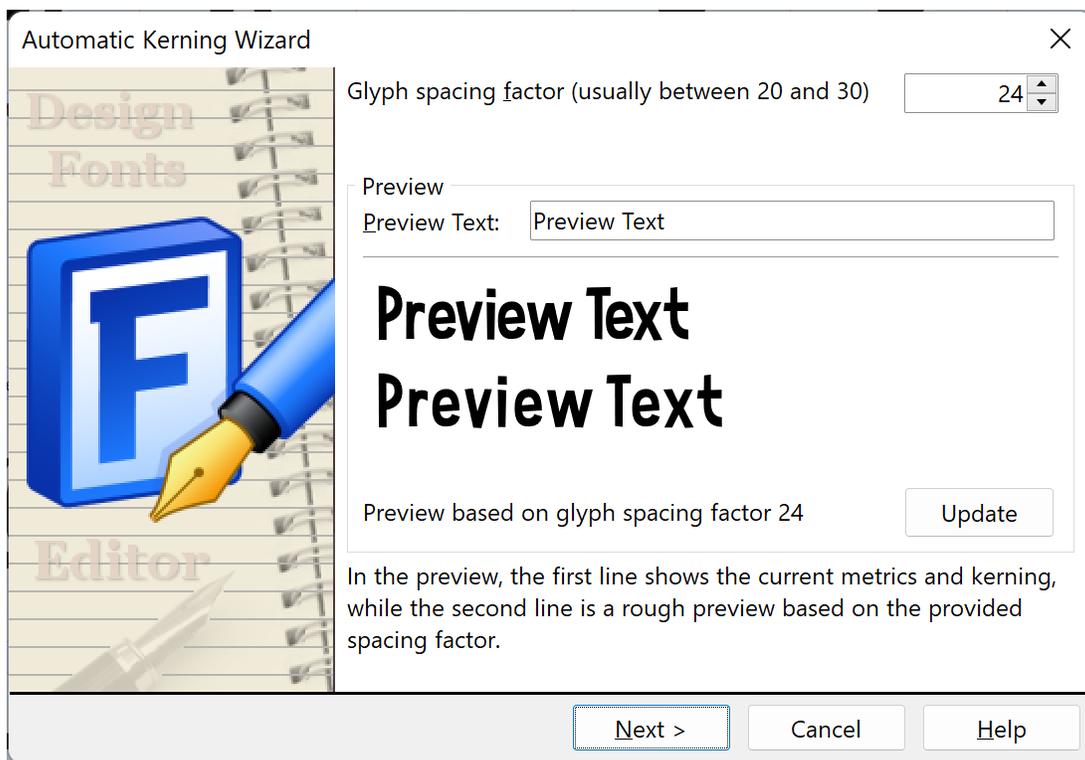
You can use the **Preview** area at the bottom of the **OpenType Designer** dialog to test the kerning pairs. You can also install the font and use an application that supports kerning.

Tip: The [Test Font window](#), which temporarily installs the font, also supports OpenType kerning. The name of the temporarily installed font looks like FontName 012345. So while the Test Font window is open, you will be able to use the font in any application (e.g. Word). However, when you open the Test Font window again you'll have to change the font's name in the application, because the temporary font name identification number always changes.

In Microsoft Office Word, select Font from the Format menu and select the Character Spacing tab. There you can turn on kerning in Word by checking the kerning for fonts field.

Note: Kerning in Symbol fonts won't be used in Microsoft Word.

Warning: Automatic Kerning can't be used with symbol fonts.



Glyph spacing factor allows you to define the distance between glyphs. The larger the factor the more space between glyphs, thus the larger the left and right side-bearings.

Preview is where you can define a preview text sample which will be shown in the preview area.

The **Next** button takes you to the next dialog where you can [set additional options](#).

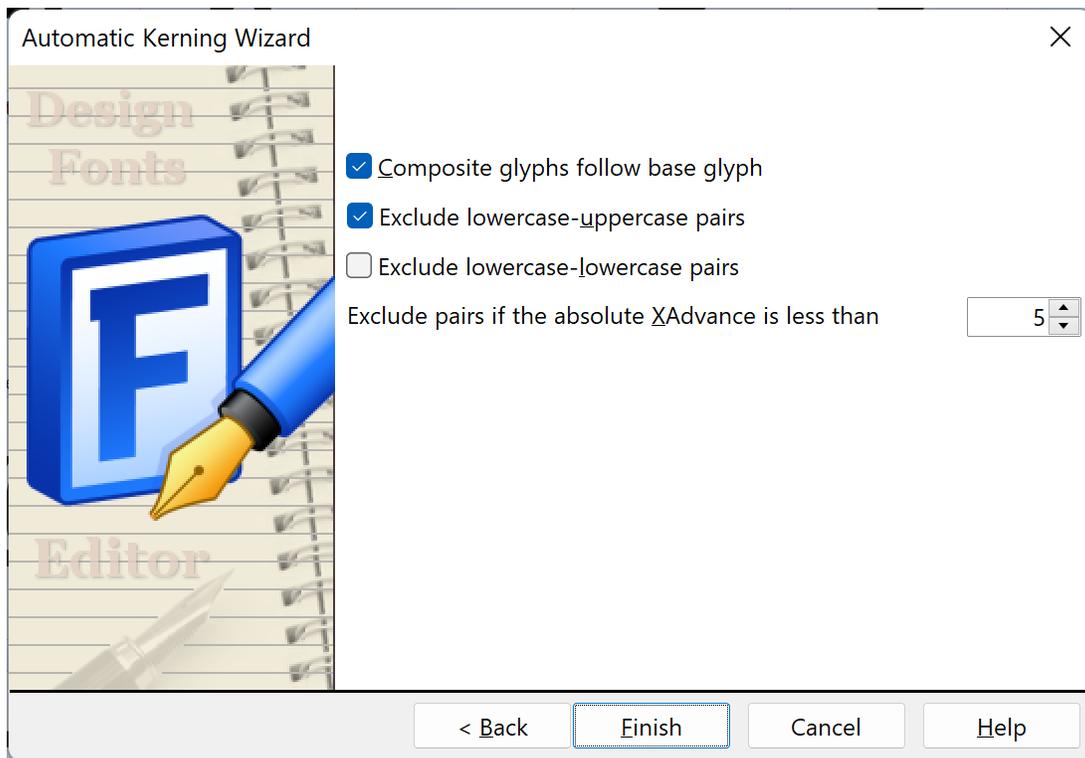
Note: The Automatic Kerning wizard is not available in the Home Edition of FontCreator.

See also:

[Autokern existing pair adjustment lookup](#)

6.8.2 Kern

In this step, enter the values for white space between glyphs and the minimum absolute kerning value.



The **Composite glyphs follow based glyph** option will ensure all composites will have the same kern values as the base glyph.

Select **Exclude lowercase-uppercase pairs** if you don't want to generate pairs for LC-UC combinations (for example aB and mE).

Select **Exclude lowercase-lowercase pairs** if you don't want to generate pairs for LC-LC combinations (for example ab and me).

You can **exclude pairs** if the kern value is below a certain threshold value.

Click the **Finish** button to start generating kerning pairs, otherwise click the **Back** or **Cancel** button.

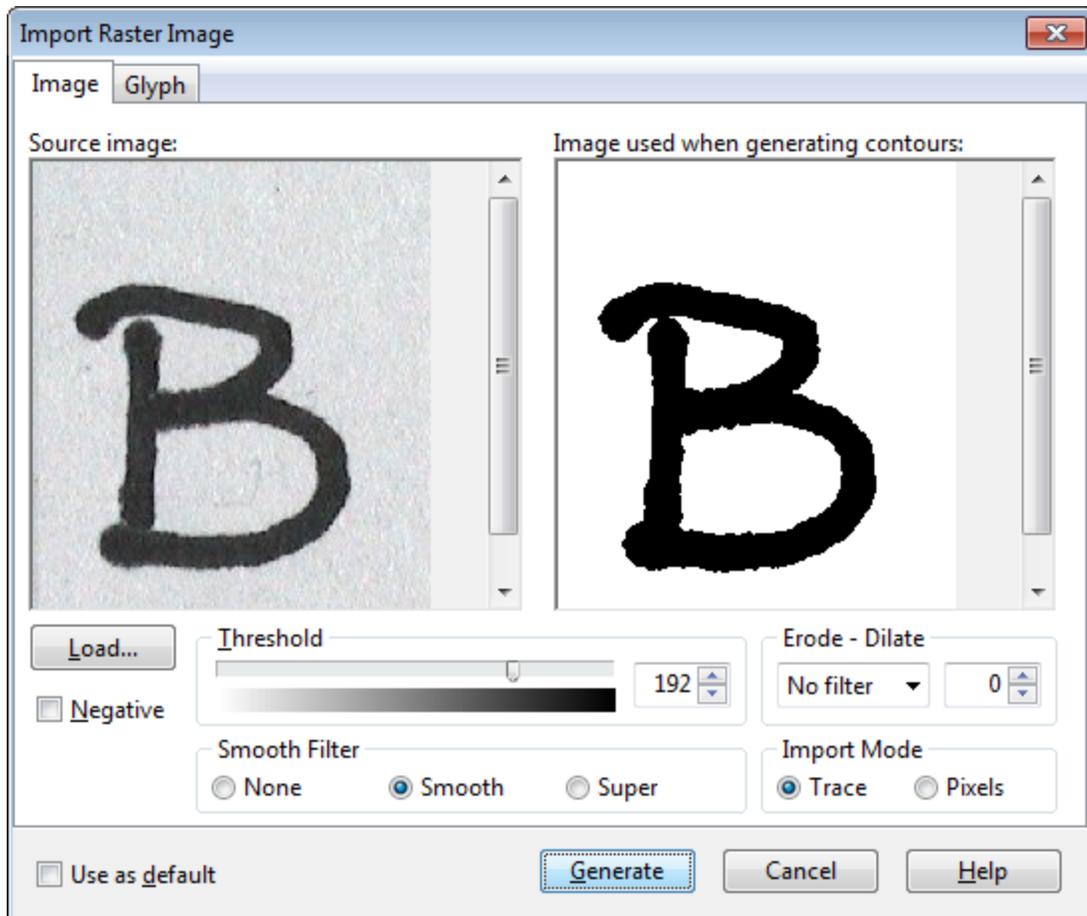
As soon as the task is completed, a new pair adjustment lookup is added to your existing OpenType layout features. You can manually make adjustments to specific kern pair values, and add and remove kerning pairs if needed.

Note: The Automatic Kerning wizard is not available in the Home Edition of FontCreator.

6.9 Import Images

6.9.1 Import Raster Image

Import Image can be selected from the **Tools** menu when you have activated a **Glyph** panel or it can be selected from the context menu after right-clicking in the panel.



When you click on the **Load** button you get an open dialog box where you can open an image file (recommended image dimension between 250x250 and 500x500 pixels). The image will be displayed on the left and there will be a bitmap that is going to be used for the conversion on the right. There are some filters and other operations you can apply to the source image before starting the conversion.

The maximum image dimension is limited to 8192x8192.

If your source image is small, a warning dialog will advise resizing the image. Select a percentage up to 400% or use a larger image if one is available.

The **Threshold** level is used to convert a color image into a black and white image. The Threshold level is the lightness value above which colors are turned black. All colors with lightness values above the level are turned into black. At a Threshold level of 1, all colors except white change to black.

Check the **Negative** image field to invert the image.

Dilation causes objects to grow in size and erosion causes objects to shrink. The amount that they grow or shrink depends on the value specified on the right of the selection box.

Use the **Smooth Filter** to smooth the image. This will usually reduce the number of generated points.

When the **Import Mode** is set to Trace, the image will be converted with curves. This is the recommended setting. In case you don't want curves (especially useful for bar code fonts and pixel fonts), set the Import Mode to Pixels.

Check the Default field to save the current settings as the default for each new import. These settings are also used when pasting an image from the clipboard. Choosing to press the Cancel button will retain these default settings.

Use the settings on the **Glyph** tab to position and resize the generated contours.

When you press the **Generate** button the conversion will start.

Tip: You can also paste an image from the clipboard or drag and drop image(s) from explorer into the Font and Glyph panels.

Note: You can't import images into composite glyphs.

6.9.2 Import Vector Image

Import Image can be selected from the **Tools** menu when you have activated a **Glyph** panel or it can be selected from the context menu after right-clicking in the panel. When you open a vector based image file, the file will be instantly imported.

In vector based image editing software you can use all kinds of objects to create your images. Such objects can be paths, lines, shapes, text, etc. That's not it, as you can also apply specific strokes to each of these objects. These strokes control thickness, how segments join, and the appearance of both ends of an open path. There are numerous other capabilities like fill objects, gradients, etc.

FontCreator can only extract the bare paths, thus completely ignores the strokes, fills, etc. And since glyphs can only consist of closed contours, all open paths are automatically closed.

Tip: FontCreator allows you to copy and paste vector based artwork from several applications like Adobe Illustrator, Adobe Photoshop, Inkscape, and Affinity Designer.

Import Settings

Scale and position depend on the [import settings](#), so check those if your imported image is too small, too large, or placed outside the visible glyph area.

Tip: Most vector editing software uses cubic bezier curves while FontCreator supports both quadratic and cubic bezier curves. FontCreator will convert imported outlines to the outline format as defined in the Desktop (ttf/otf) [Font Export Settings](#). While designing your font it might be wise to set this outline format to match the format used within your vector editing software, as then copy and paste back and forth doesn't require any conversions.

Importing Portable Document Format (PDF) and Scalable Vector Graphics (SVG)

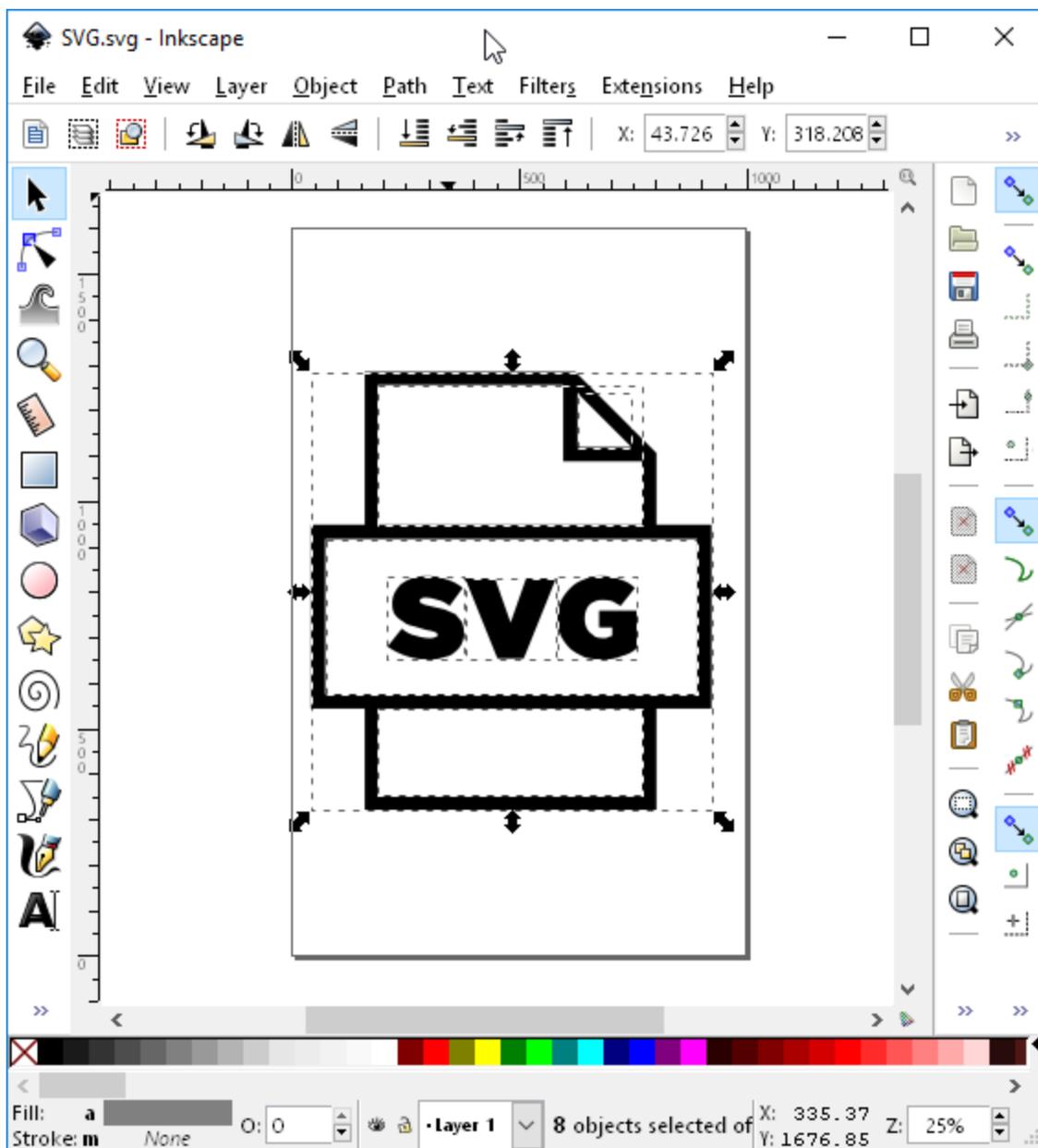
FontCreator supports path, rectangle, circle, and ellipse, so no text, fills, gradients, etc.

If, for some reason, the glyph outline seems wrong, ensure that your original SVG doesn't contain overlapping paths.

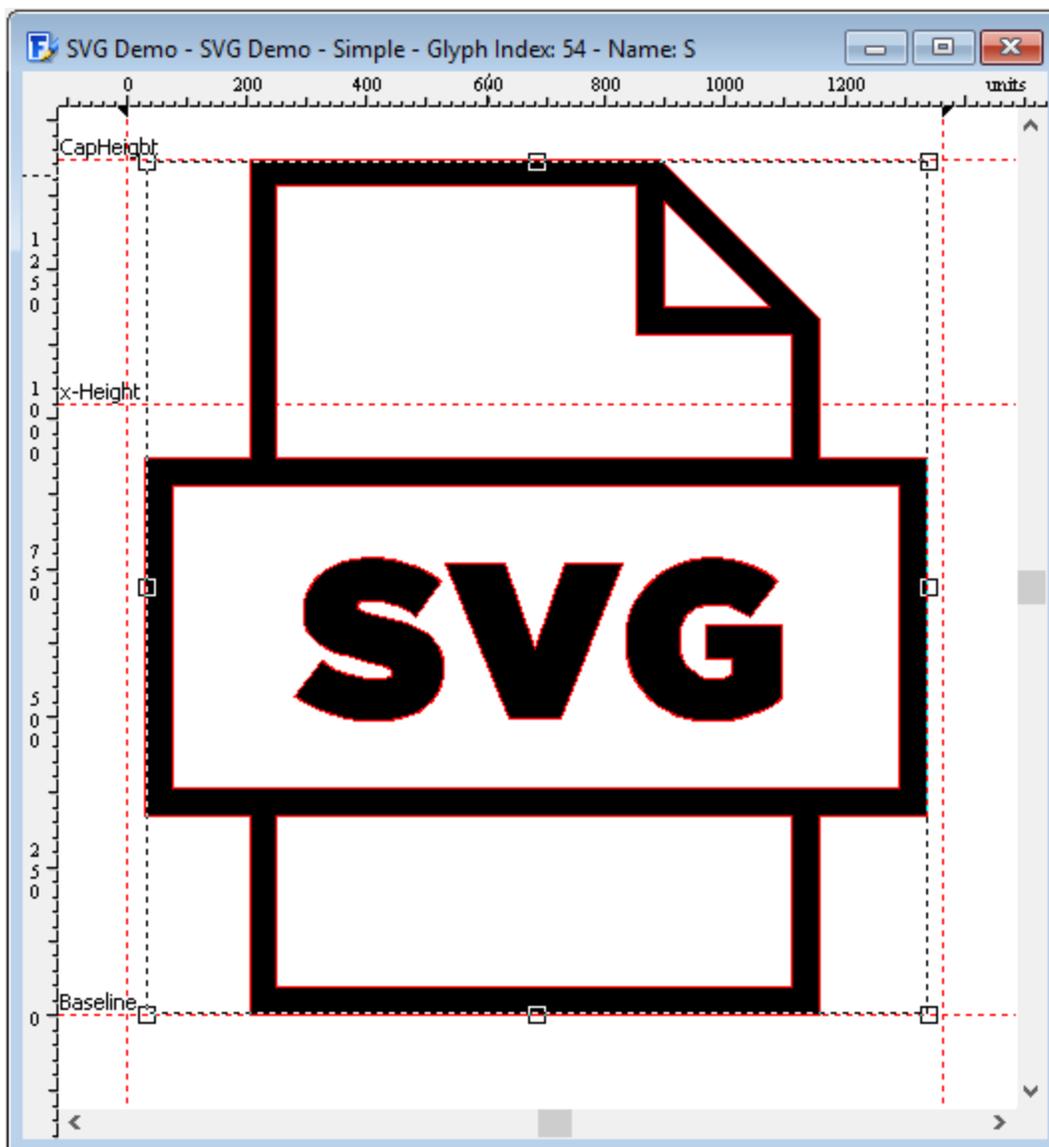
Importing from Inkscape

For best results ensure that all strokes are converted to closed paths within Inkscape:

- select all objects
- select Ungroup from the Object menu
- again select Ungroup from the Object menu as some groups might reside in other groups, etc.
- select Stroke to Path from the Path menu



Then copy and paste that image in FontCreator.



Tip: you can also copy outlines from FontCreator to the clipboard and paste them in Inkscape. To paste in Inkscape, go to the main menu and select File -> Paste In Place (or Ctrl+Alt+V). This ensures the position remains the same between the two apps.

Importing from Affinity Designer

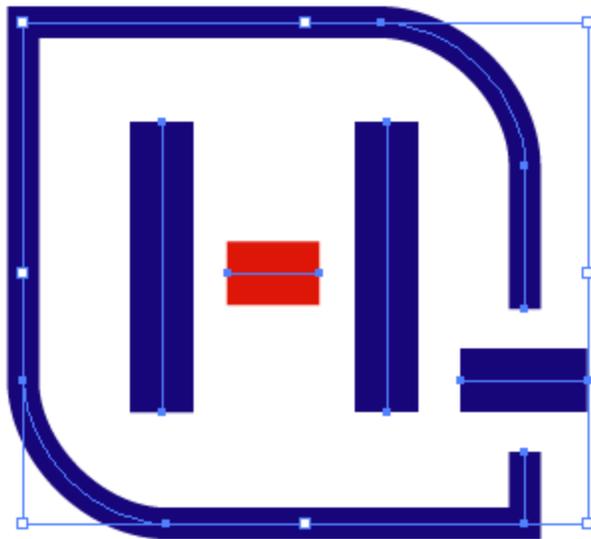
This mostly works similar to what is explained above with Inkscape.

When pasting outlines into Affinity Designer, it tries to be smart and centers the outline. Then when you paste back into FontCreator, it also scales based on the

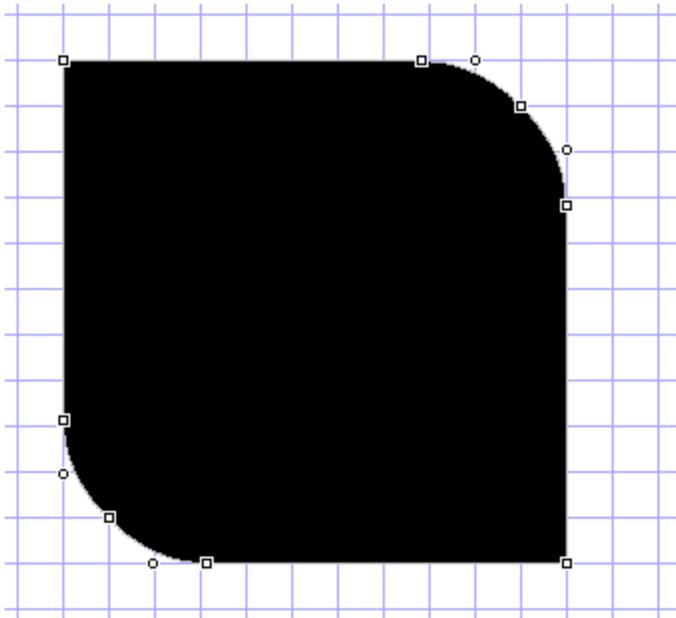
documents DPI setting. To avoid scaling, you need to set it to 72 in the Affinity document.

Importing from Adobe Illustrator

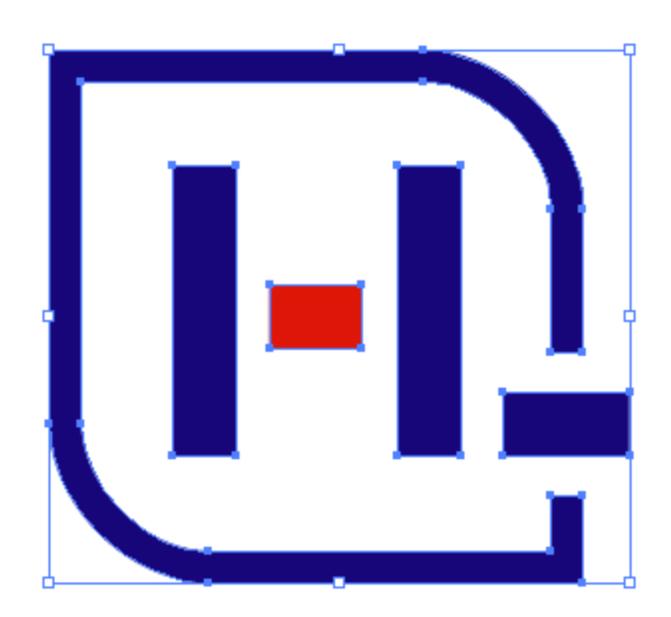
Here is a logo created with Adobe Illustrator. It contains four lines and a rectangle with two rounded corners and a gap at the lower right. All of these objects contain strokes to give them thickness.



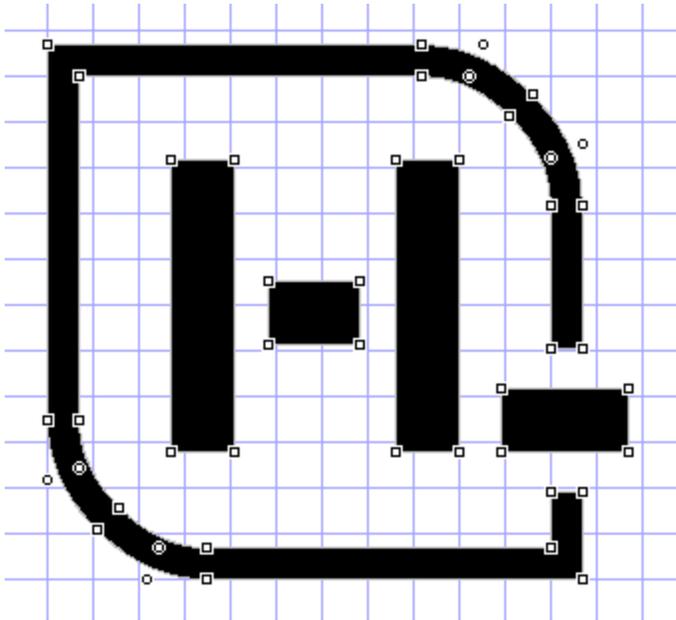
When imported into FontCreator, the result is not as intended as the strokes are not imported, thus the thickness of the objects is stripped off. The lines are all discarded as contours with only two points are useless. The rectangle is imported but the gap is no longer there.



One easy step ensures that our vector image can be safely imported into FontCreator. In Adobe Illustrator, select all objects, then from the main menu select Object -> Path -> Outline Stroke. The result is shown here:



When we import this version of the vector image the result is a perfect fit!



5 contours with 44 points is the final result.

Tip: It is also best to immediately resize the imported contours right after you've pasted them into FontCreator. The "Add on-curve extremes" feature will round the coordinates to whole integers, so always do this after you've made sure that the size of the imported contours is correct.

Tip: You can also paste an image from the clipboard or drag and drop image(s) from explorer into the **Font** and **Glyph** panels.

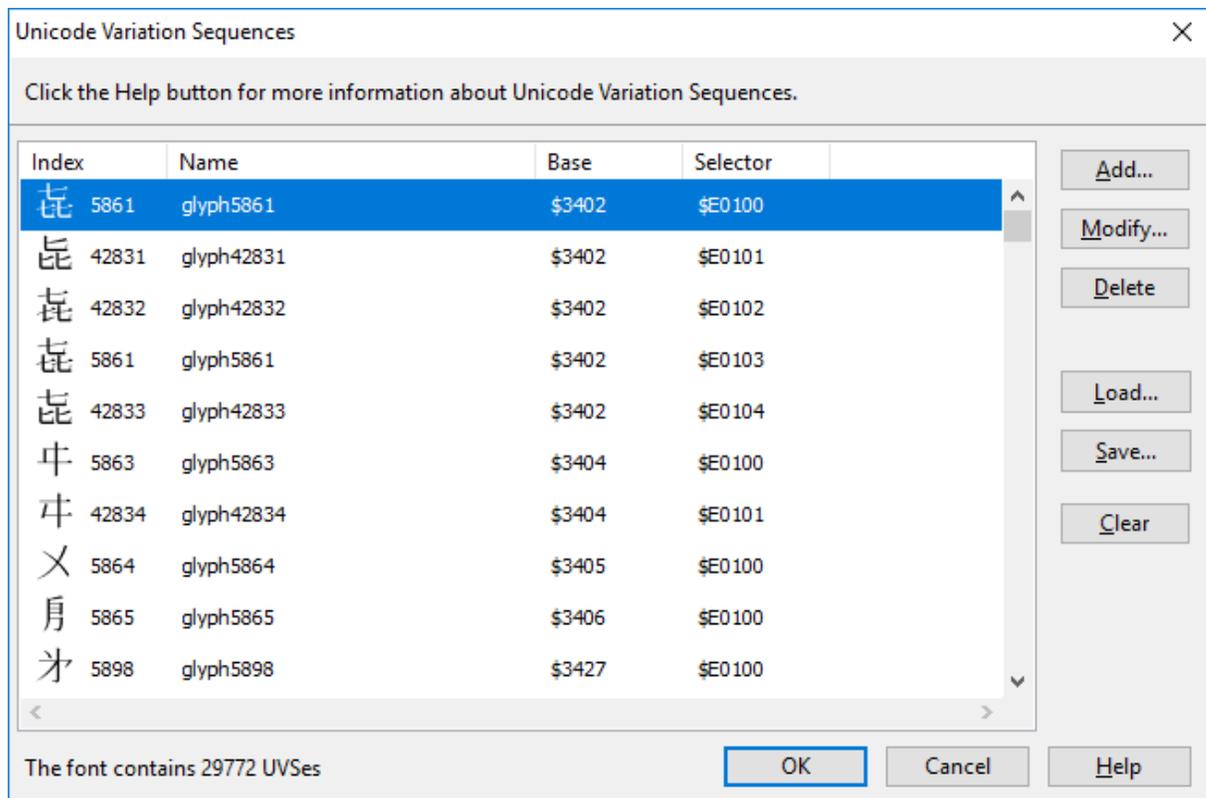
Note: You can't import images into composite glyphs. If you want to include more contours, then you need to add them to another glyph and add it as glyph member, or convert the glyph to a simple glyph.

6.10 Unicode Variation Sequences

A Unicode variation sequence is a special kind of character substitution. Variation selectors provide a mechanism for specifying a restriction on the set of glyphs that are used to represent a particular character. They also provide a mechanism for specifying variants that have essentially the same semantics but substantially different ranges of glyphs.

A variation sequence consists of a base character (or a spacing mark) followed by a single variation selector. The Unicode Consortium maintains three lists with variation sequences:

- Standardized variation sequences.
- Emoji variation sequences. The variation selector character is either U+FE0E (text style) or U+FE0F (emoji style).
- Ideographic variation sequences. The variation selector character is in the range U+E0100 to U+E01EF.



Import and Export

You can import and export Unicode variation sequences, using this format:

Base Selector GlyphName

for example:

\$3515 \$FE00 CJKCOMPATIBILITYIDEOGRAPH-2F824

Note: User defined variation sequences are not valid. If you are in need of a custom substitution, then you can most likely use glyph substitutions which are part of OpenType layout features.

More information about variation sequences and the lists can be found here:

<http://unicode.org/faq/vs.html> 

6.11 Sorting Glyphs

To change the order of the glyphs within a font select one of the options from the Sort Glyphs submenu in the Tools menu.

With all but the From File option, these glyphs (if available) will always come first in the new order.

- .notdef
- .null
- nonmarkingreturn
- space

Note that the .null and nonmarkingreturn glyphs are no longer required, but it remains important to have the .notdef glyph as first glyph. More information about these special glyphs can be found here: [Recommended Glyphs](#).

Design mode

Glyphs will be sorted by these rules:

- Glyphs are grouped together, starting with capital letters, then lowercase, etc. Each group is followed by the accompanying alternate forms.

Unicode code-points

Glyphs will be sorted by these rules:

- Glyphs are sorted by their code-points

Glyph names

Glyphs will be sorted by these rules:

- The glyphs are sorted by their glyph names

Glyph type (empty, simple, composite)

Glyphs will be sorted by these rules:

- The glyphs are sorted by their glyph type
- Then all remaining glyphs are sorted by their glyph names

Alphanumeric

Glyphs will be sorted by these rules:

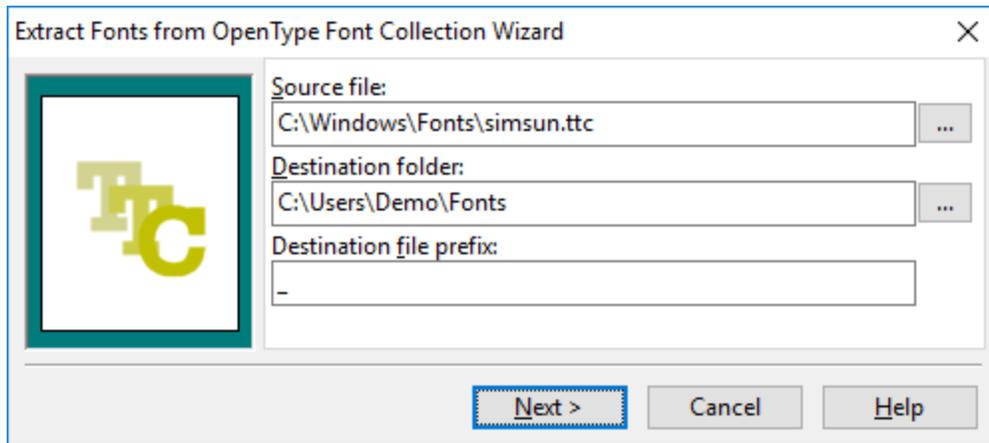
- The glyphs are sorted by their mapped character. This feature uses default compare functions as used in Windows.

From File

Glyphs will be sorted by the order of the glyph names. You can either use a plain text file; per line one glyph name. You can also provide a UFO lib.plist file.

6.12 OpenType Collection

An OpenType Collection (formerly known as TrueType Collection) file is one or more OpenType fonts combined into one file. The **Extract Fonts from OpenType Collection Wizard**, available from the **Tools** menu, can extract those fonts.



Source file

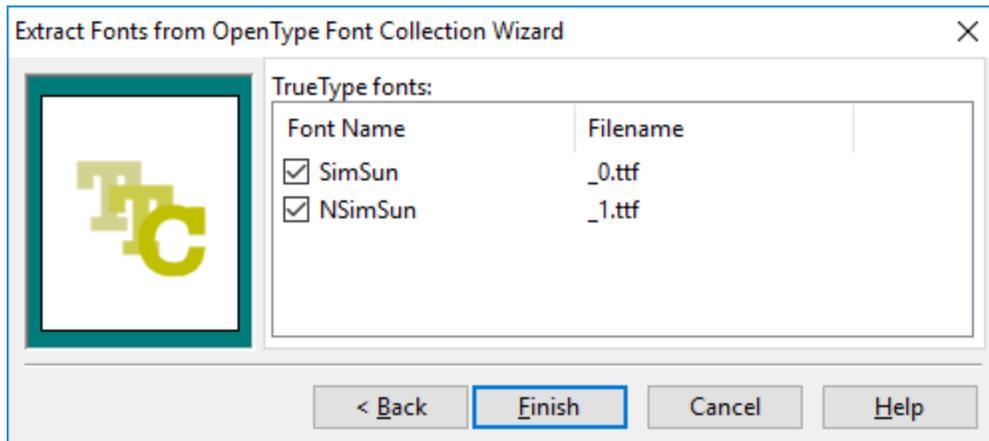
Select the OpenType Collection file.

Destination folder

The extracted font files will be saved in this folder.

Destination file prefix

The extracted files start with this prefix.



Here you will see all fonts available in the OpenType Collection. Select the fonts you want to extract and press the **Finish** button.

6.13 External Tools

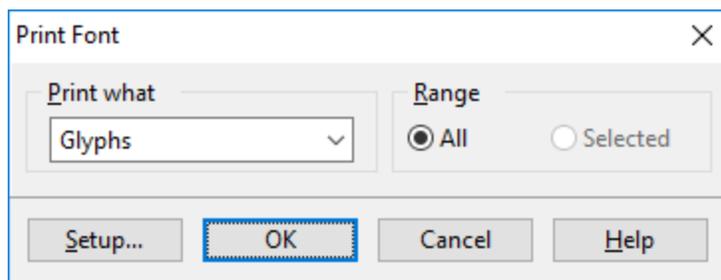
To quickly access Windows Fonts folder, Character Map and [MainType](#) (if available), select **Launch Externals** from the **Tools** menu and select the tool you want to open. Optionally you can define up to three external tools.

For each of the three user definable external tools, you need to specify a title, which will be shown in the **Launch Externals** submenu, and the location of the application's executable. You can include an ampersand in the title, which will be shown as an access key in the menu (if access keys are enabled in Windows).

6.14 Printing

6.14.1 Print Font

This option (select **Print** in the **File** menu) is available when the **Font** panel is active. If a **Glyph** panel is active you will be able to [print a glyph](#).



You can choose what kind of font information you want to print:

- Glyphs
- Properties

Print Glyphs

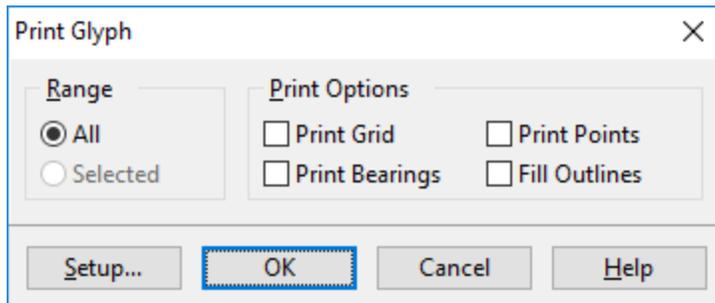
Print all or the selected glyphs.

Print Properties

Print all fields from the [Font Information window](#).

6.14.2 Print Glyph

Select **Print** in the **File** menu to print a single glyph. This option is available when a **Glyph** panel is active. If one or more contours are selected, only the selected contours can be printed. If the **Font** panel is active you will be able to [print the font](#).

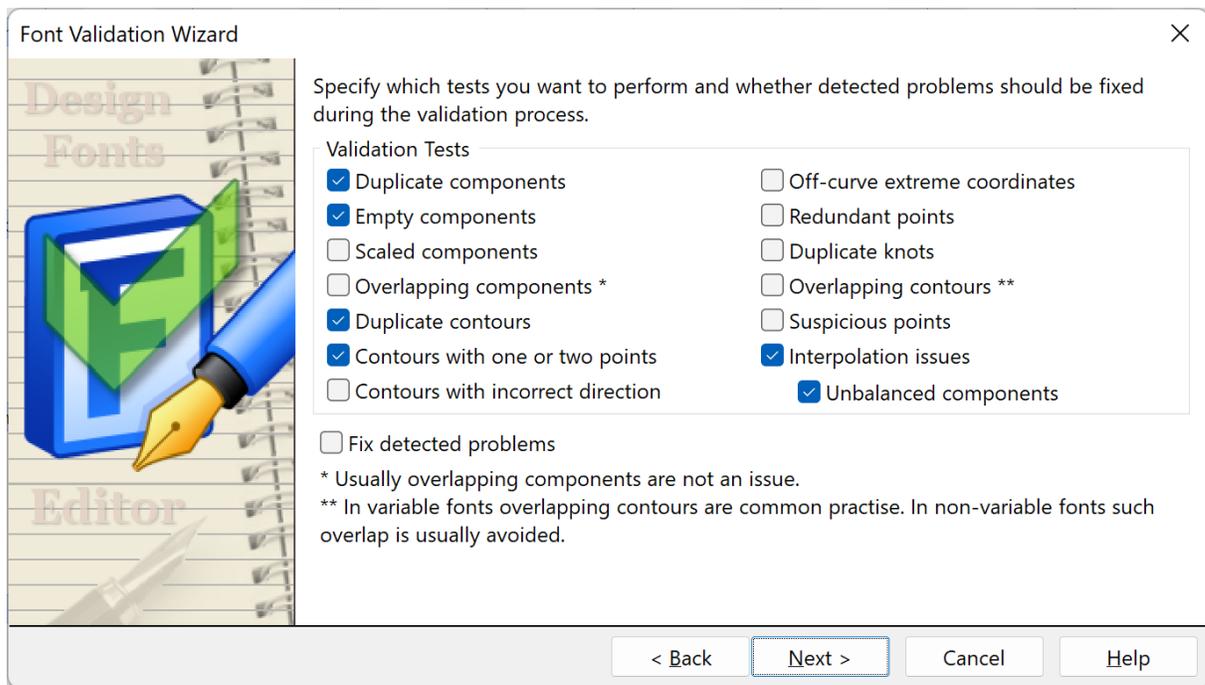


The print options allow you to print the grid, points and side-bearings, and you can choose to fill the outlines.

6.15 Font Validation

6.15.1 Setup

Designing glyphs can be very complicated. The **Font Validation** wizard, available from the **Font** menu, identifies common potential problems and if possible points you to the specific item (e.g. glyphs, contours and coordinates). It validates all glyphs and optionally fixes detected problems.



The specific validation tests are explained here:

Duplicate components

This problem will be reported when validating composite glyphs with two or more identical glyph members.

Empty components

This problem will be reported when validating composite glyphs with empty glyph members.

Overlapping components

This problem will be reported when validating composite glyphs with intersecting glyph members. Some font designers consider overlapping composite components an error, while others don't mind. We recommend to avoid overlapping contours.

Duplicate contours

This problem will be reported when validating simple glyphs with two or more identical contours.

Contours with one or two points

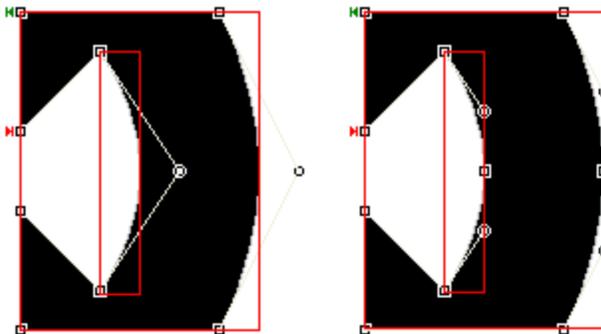
This problem will be reported when validating simple glyphs with contours with one or two points.

Contours with incorrect direction

This problem will be reported when validating simple glyphs with contours that have an incorrect direction (Contours that need to be filled black must have a clockwise direction. If we want to make a white area inside an existing contour we must make the direction of the new contour counterclockwise.).

Off-curve extreme coordinates

This test checks whether all off-curve points are inside the global bounding box (and optionally their local bounding box). You can customize this feature through the [Options dialog](#).



The left image shows two red rectangles. The large one is the global bounding box and the smaller rectangle is a local bounding box between an on-off-on-curve sequence. Both off-curve points lie outside their bounding box. With local detection enabled, the right image shows the result of clicking the **Add on-curve extremes** button on the **Validation** panel.

Redundant points

This problem will be reported when validating simple glyphs with contours with redundant points.

Duplicate knots

This problem will be reported when validating simple glyphs with contours with two adjacent points that have the same coordinates but one is on-curve and the other is off-curve.

Intersecting coordinates

This problem will be reported when validating simple glyphs with (self-)intersecting contours. Avoid crossing contours whenever possible.

Note: Older PostScript Level 2 drivers do not support overlapping contours.

Suspicious points

This problem will be reported when validating simple glyphs with contours that have points that require attention because they are likely to be incorrectly positioned.

Interpolation issues

Detects these issues with variable fonts:

- Mixed up contours and/or glyph members
- Start points not matching
- Different contour directions

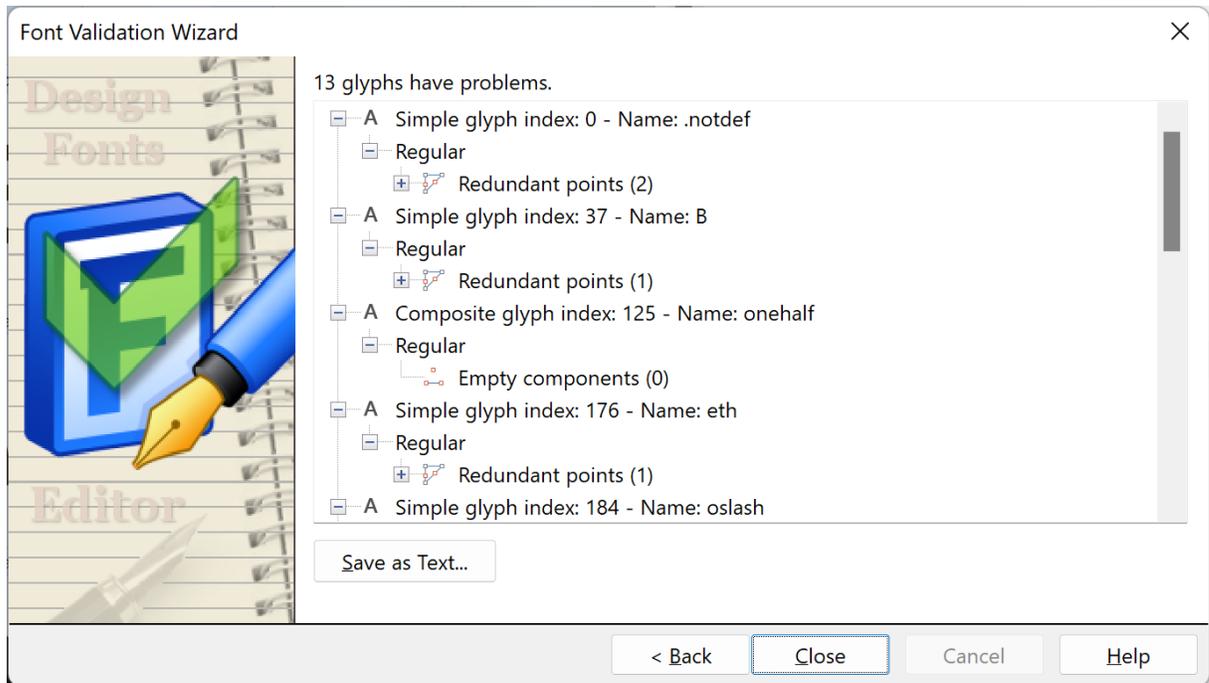
Unbalanced Components

This issue is detected when in a variable font a glyph uses components that have different scaling or rotating values. This is not an issue during the design, but as the OpenType format does not support it, such glyph outlines will be converted to simple outlines (contours) on export.

Note: The Validation features are not available in the Home Edition of FontCreator.

6.15.2 Results

After the validation process the (remaining) problems are shown for all glyphs. Optionally the report can be saved.



These glyphs will also become available through a special “Validation Issues” category within the Font panel. While this category is selected you can go to previous and next glyph within a glyph panel to see them step-by-step.

Note: The **Validation** features are not available in the Home Edition of FontCreator.

6.16 Testing and Installing Fonts

6.16.1 Preview in FontCreator

There are several places in FontCreator that allow you to preview your font, the most obvious places are the [Preview panel](#), the preview area in the [OpenType Designer](#), but you can also use the Text tool within a glyph panel.

On the upper right side there are two fields which allow you to set the size of the text and the text itself. To force a line-break in the preview area, enter “/newline” in the preview text.

The preview is not a full blown OpenType text shaping engine, but is still extremely useful while testing your font, metrics, kerning and other OpenType layout features it contains.

Shaping Engine

The preview is not a full blown OpenType text shaping engine, but is still extremely useful while testing your font, metrics, kerning and other OpenType layout features it contains. Use [Test Desktop Font](#) to see how your font behaves in Windows, and use [Test Web Font](#) to test it in your default Web browser.

You can choose which features need to be active, so only those will be used to shape the text within the preview area. You can also choose to use a more advanced **Shaping Engine**, which behaves more like other software like Microsoft Word, Adobe InDesign, Serif Affinity Designer, and web browsers. To enable it, check the feature named “_shaper”. This will trigger (basic) normalization and will then process specific features in a fixed order, even if they are not selected, and after that it will process all remaining active features.

Note: The engine first splits the provided text into runs, so each run contains a different script. Therefore OpenType layout features won't work between different scripts.

The shaping engine will process the following features for glyph substitution:

locl, ccmp, nukt, akhn, rphf, pref, rkrf, abvf, blwf, half, pstf, vatu, cjct, isol, init, medi, fina, abvs, blws, calt, clig, haln, liga, pres, psts, rclt, rlig, vert, vrt2.

and these for glyph positioning:

curs, dist, kern, mark, abvm, blwm, mkmk.

Some of the above features will only be applied if the specific glyphs do match the meaning of the feature. For example isol will only be processed if the input glyph is in an isolated form within the given text.

Tip: use the [proofing window](#) to analyze and inspect OpenType layout features that apply to the current preview text.

Note: if the _shaper is enabled, it will continue to process the features as explained earlier in this paragraph.

Preview Text

You can type a line of text, but you can also use special formatting to include unmapped glyphs, new line, etc as discussed below.

Include Glyph by Name

You can type a slash followed by a glyph name, which is especially useful when a glyph is unmapped. You can add a space or backslash immediately after it, so the next character can be interpreted as a next letter.

For example `/one/two\a` will show 12a

Include Glyph by Code-point

You can type a slash followed by a Unicode code-point in either hexadecimal or decimal notation. So for the @ sign you can type these:

`/$40`

`/64`

Include Slash

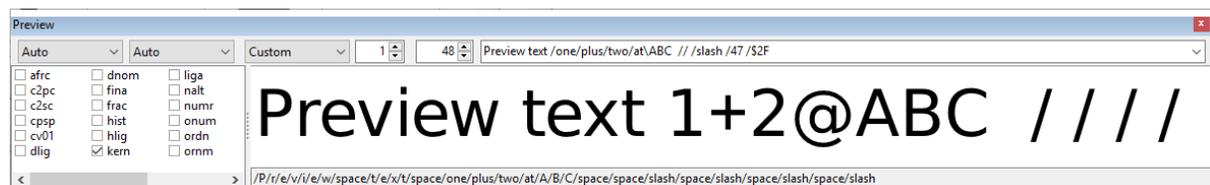
It requires some special attention to get a slash in the preview, as the slash is used to add a glyph by name and code-point. You can either type two slashes, or use glyph by name or glyph by code-point, so all of these should work:

`//`

`/slash`

`/47`

`/$2F`



Force New Line

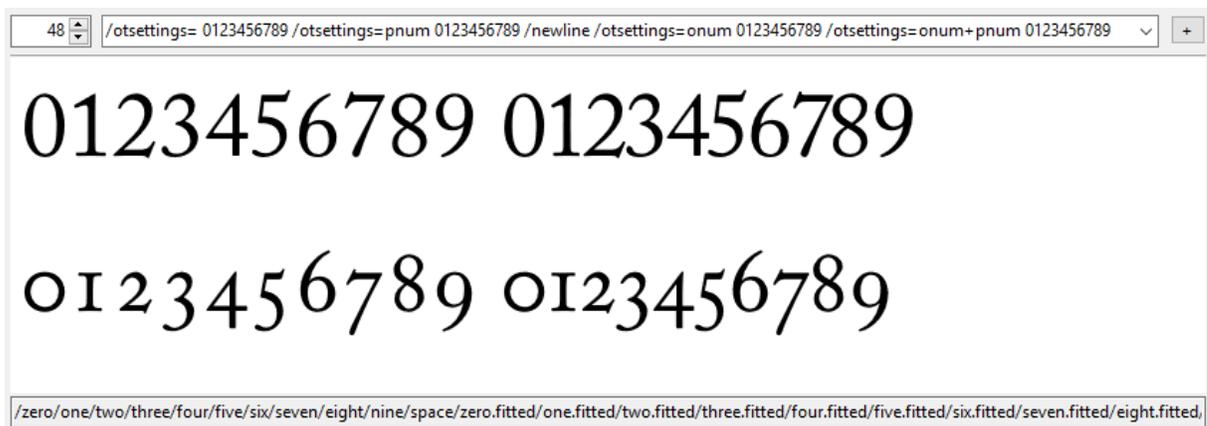
To force a line-break in the preview area, enter `"/newline"` in the preview text.

Activate features for parts of the preview text

The preview text allows you to enable and disable features for subsequent text. This allows you to test several feature combinations at once, so for example you can compare how small capitals behave with and without kerning applied. It also allows you to compare the way lining and oldstyle figures work with tnum or pnum enabled.

Within the preview text provide `"/otsettings=` to override or `"/otsettings+` to enable and disable features. For example:

```
/otsettings= 0123456789 /otsettings=pnum 0123456789 /newline /otsettings=onum
0123456789 /otsettings=onum+pnum 0123456789
```



6.16.2 MainType

MainType is a powerful font manager for Windows that helps you maintain your fonts. Although its main purpose is installing and uninstalling fonts, it also allows you to insert special characters into documents and the Test window.

You can download MainType from here:

<https://www.high-logic.com/font-manager/maintype> 

6.16.3 International Keyboard

Instead of memorizing a long list of Alt number combinations, or using the Character Map, you could change your language and layout from English-US to United States-International (or any other available language on your system).

Here's how:

- Single-click the Start menu, mouse over Settings, and then single-click the Control Panel.
- Double-click the Keyboard icon and then click the Language tab
- Click Add.
- Under Keyboard layout, place a checkmark next to United States-International.
- Click OK.
- Click Apply. You may be asked to insert your Windows system disk to finish loading the process.

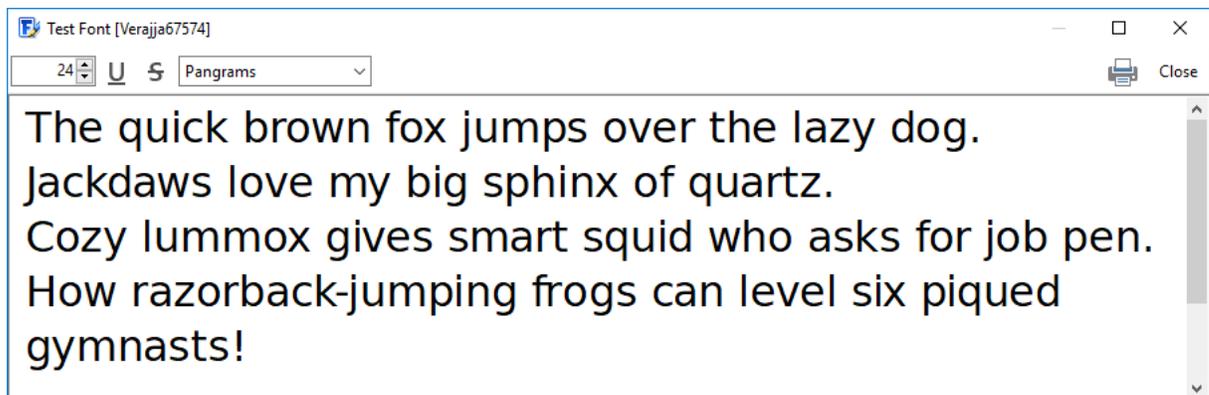
You can type in other languages without knowing the Alt codes for a non-English alphabet. For example, type ~ followed by N to get Ñ. A tilde followed by any letter will place the appropriate accent on the letter you choose.

6.16.4 Test Your Font

6.16.4.1 Test Desktop Font

Test Desktop Font (ttf/otf)

If you want to know how your font is going to look you can test it any time during the development process. Choose Test Desktop Font (ttf/otf)... from the Font menu or use the shortcut F5.



You can enter your own text into the Test window. FontCreator will remember this text so you can always test your fonts with your preferred text.

If you want to test how your font will look when it's printed you press the Print button.

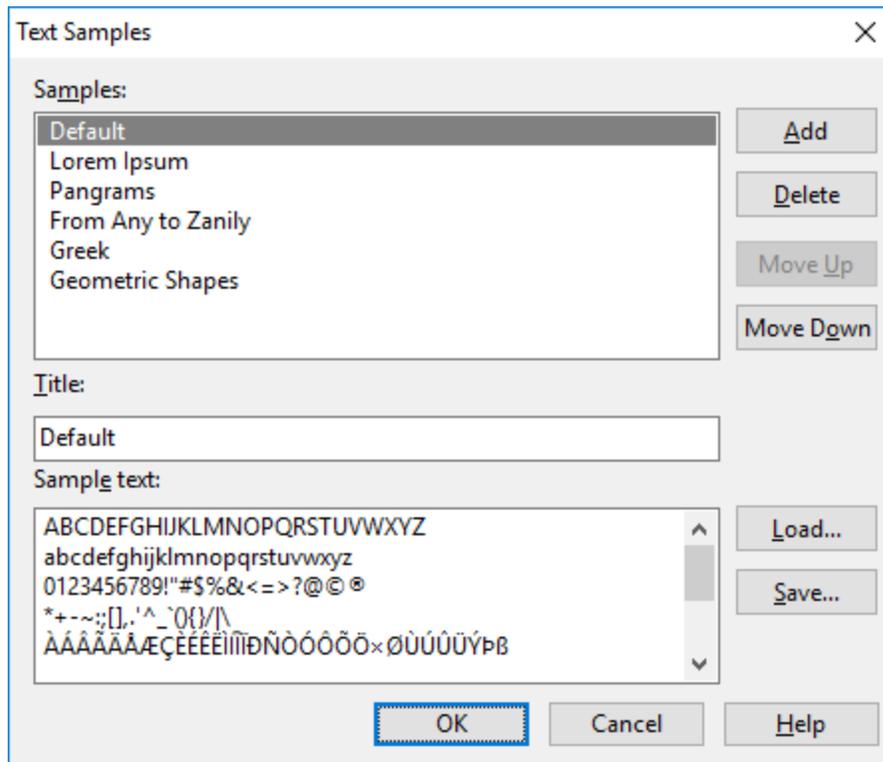
Test in other Software

While this Test Font dialog is open, you can also use your font in other software. The font has a temporary name as shown between brackets in the window caption bar, which is usually the original font name followed by a number.

Word 2010 and up do support several OpenType features, like ligatures and contextual alternates, but you will manually have to enable such support in your documents. Within Word do open the Font dialog, and select the Advanced tab. Then check the options you want to enable, e.g. "Standard Ligatures" and/or "Use Contextual Alternates" and click OK.

6.16.4.2 Test Font - Edit Text Samples

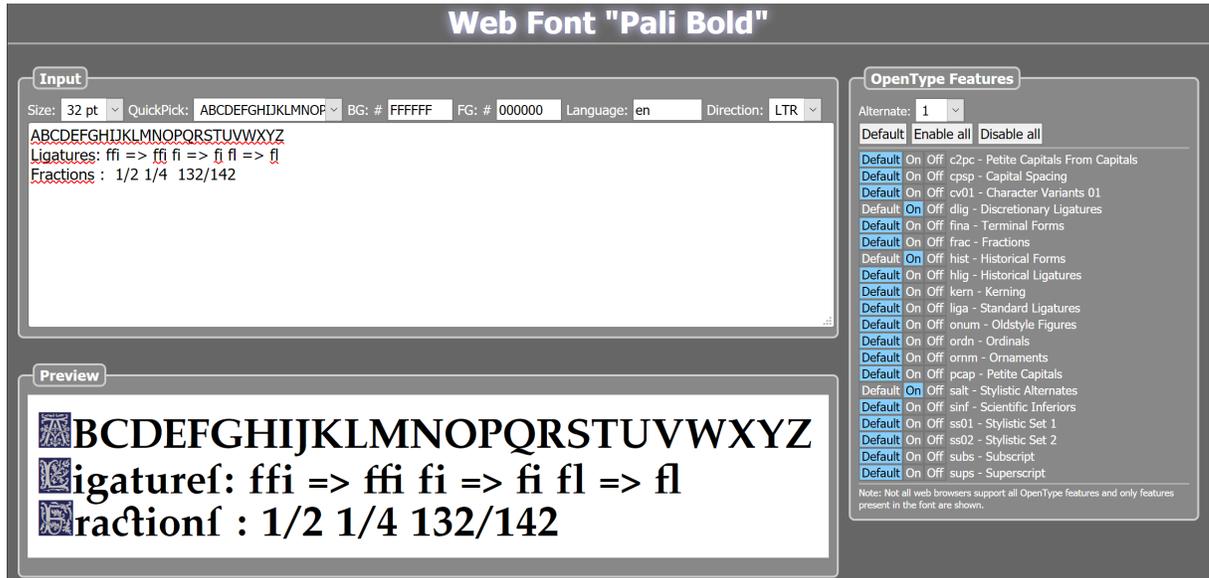
You can edit the text samples used on the Test Desktop Font dialog by right-clicking on the test area and select **Edit Text Samples**.



Here you can add, delete and change the order of the text-samples.

6.16.4.3 Test Web Font

You can also test your font and **OpenType layout features** as a web font on a locally generated web page. Choose **Test Web Font (woff/woff2)** from the **Font** menu or use the shortcut **Ctrl+F5**



If OpenType Layout Features are present, you can activate them using the buttons (default, on, and off) on the right.

If you intend to use your font on the Web, ligatures will be disabled as soon as you set letter-spacing to a non-default value. For example:

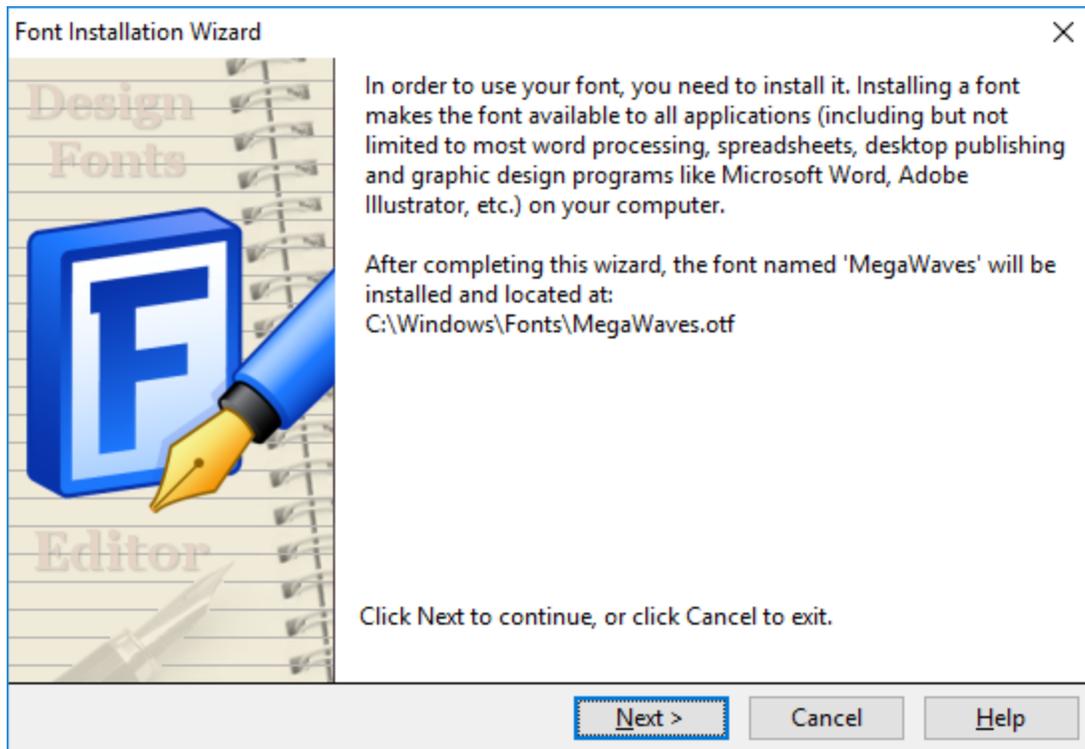
letter-spacing: 0.1em;

You can edit the predefined preview text through the [FontCreator Data Files](#).

6.16.5 Installing Fonts

Although it is possible to install a font through Windows fonts folder, FontCreator has its own **Font Installation** wizard. To make your font available to other applications select **Install** in the **Font** menu.

The **Font Installation** wizard will guide you through the installation process. At the end of the installation process you will be informed that the font is installed successfully. Now you will be able to select the font in any program that supports fonts.



Note: If you are reinstalling the font, it is recommended you delete the font BEFORE installing the new version.

Note: Don't just drop the font into the Windows Fonts folder!

6.16.6 Character Map

With Windows Character Map you can insert special characters into documents and the Test window.

From your Start menu mouse over Programs, Accessories, and System Tools, and then click on Character Map. In the Character Map select the appropriate font. Click the letter or symbol you need and click the Select button. Add more characters if

needed and finally click the Copy button. Now go to your document and select Paste from the Edit menu or use the shortcut Ctrl-V.

Note: To type special characters (like the copyright sign) of the font in your word processor or page layout program, hold down the Alt key, and then, by using the numeric keypad, type 0 (zero) followed by the corresponding decimal character code. Make sure NUM LOCK is on.

Another way to input Unicode characters is the hexadecimal entry method that works with WordPad and Microsoft Word. Probably more applications will follow soon. Basically you type a character's hexadecimal code (in ASCII), making corrections if needed, and then type Alt+X. This will replace the hexadecimal code by the corresponding Unicode character. The same key combination Alt-X can be used to reveal a character's code. If the hexadecimal code is preceded by one or more hexadecimal digits, you need to select the code so that the preceding hexadecimal characters aren't included in the code. The code can range up to the value 0x10FFFF, which is the highest character in the 17 planes of Unicode.

Part



7 Toolbars and Panels

7.1 Overview

The toolbars give you one-click access to many of the commands on the menus. Some menu items have toolbar icons next to them so that you can quickly associate the command with that icon.



Toolbars

FontCreator's toolbars can be shown or hidden as needed. By default the **Standard, Tools, Drawing, Grid, Glyph, Overview** and the **Align or Distribute** toolbars are docked below the menu bar. Choose **Toolbars** from the **View** menu to select toolbars to display or hide, or right-click on any toolbar to get the same submenu. You can also lock the docked toolbars, so that you don't accidentally move them.

The documents tab bar, listing opened fonts and glyphs, cannot be moved or undocked.

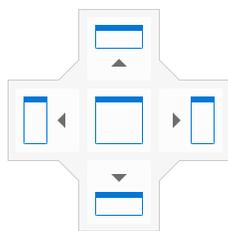
Tips: Toolbars can be rearranged by dragging, docked on the left, right, top, or bottom of the FontCreator window, or made floating. Floating toolbars can also be resized, which will progressively rearrange the icons in them from a horizontal to vertical layout. To undock a docked toolbar, double-click its grab handle where the four-arrow cursor is displayed. To dock it again, double-click the floating toolbar's title bar.

Panels

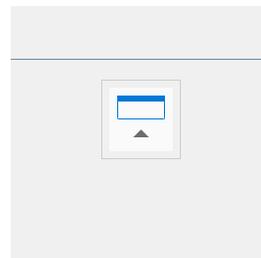
The layout of FontCreator can be modified by dragging and docking the panels. The Fonts and Glyphs panels can be closed, while the other panels can be hidden. All panels can be docked, floated, or even tabbed together to your liking. To move a panel, click and hold down the left mouse button on the caption bar of the panel, and start moving your mouse. When you start dragging one of the panels, a cross will appear on the current panel indicating where the panel can be docked. If you

hover your mouse over one of the icons, a blue rectangle will show where the panel will be docked. You can also use one of the 4 dock icons that appear on the top, left, right, and bottom of the main window; this will dock the panel across the entire width or height.

Releasing the panel on the center icon will create a set of tabs so that you do not have the panel always visible, but can easily access it by clicking the appropriate tab. To keep a panel floating above the main form simply release your mouse button while not hovering over any of the docking icons.



Dock positions available while hovering over another panel



Dock icon appearing on top of the main form while dragging a panel

The Font Properties, Masters and Layers, Variations, Glyph Properties, Transform, Validation, Preview, Background, Samples, Anchors, Palette, and Members panels can be toggled on/off with shortcut keys Ctrl+F2, Ctrl+F6, Ctrl+F7, F4, F6, F7, F8, F9, F12, Shift+F4, Shift+F2, and Shift+F8 respectively.

Undo

The **Undo** command from the **Edit** menu reverses the last action made to the active font. So using Undo returns the font to its state prior to the most recent operation.

Note: The Undo button on the toolbar has a small arrow which allows you to pull down a menu and select multiple actions to be undone.

Redo

The **Redo** command from the **Edit** menu re-applies the actions or commands on which you have used the Undo command. FontCreator supports **Multiple Redo**, which is particularly useful if you have performed the undo command more than you had intended. If this occurs, and you want to re-apply them, either choose the Redo

command as many times as necessary or use the drop arrow on the **Redo** button located on the **Standard** toolbar.

7.2 Font Properties

7.2.1 Font

The Font Properties panel contains several tabs, containing most of the font settings, like version, units per em, names, font metrics, and also information that is required for variable fonts, like axes, masters, and named instances.

General and legal information and character coverage about the font is found on the Font tab (in the **Font Properties** panel). On the **Font** menu, click **Properties**, and then click the **Font** tab.

Font Properties ✕

Font Axes (2) Masters (3) Instances (8)

General

Family Name ...

Units per Em

Version .

Created

Modified

Update modified timestamp on export (recommended)

Legal

Copyright ...

Trademark ...

Description ...

License Agreement ...

License URL ...

Embedding Licensing Rights

No subsetting

Bitmap embedding only

Vendor ID

Vendor (Manufacturer) ...

Vendor URL ...

Designer ...

Designer URL ...

Sample Text ...

Variations PostScript Prefix ...

General

Family Name

The font name. Maximum length is 31 characters.

In general you are strongly recommended to use only letters, digits and the space character. For example the Mac refuses fonts which contain a quote in the font name, so **Jack's font** will fail to work on a Mac.

For non-latin fonts, like Chinese, Japanese, Korean, Hebrew, Arabic, etc, you can add localised names, accessed by pressing the [...] button.

Note: The font style is available on the [Instances tab](#).

Units per Em

Valid range is from 16 to 16384. Nowadays 2048 units per em value is the recommended value for all fonts including large Latin or non-Latin script fonts. However, fonts with fine details (e.g. circular contours that are less than 20 funits in diameter) will benefit from larger values, e.g. 4096 or 8192 units per em.

It is common but not a requirement to have a value that is a power of 2 for fonts that have TrueType outlines.

For OpenType fonts with CFF based outlines it is recommend to set units per em to 1000.

This value is used to convert values in the pixel coordinate system by multiplying them by a scale. This scale is:

$\text{PointSize} * \text{resolution} / (72 \text{ points per inch} * \text{units_per_em})$

Note: use the [Glyph Transformer](#) if you also want to resize glyph outlines.

Version

The version of the font, usually the initial release starts with version 1.0.

Note: For historical reasons, the Font revision version is not used by Windows to determine the version of a font. Instead, Windows evaluates the version string from the Version String field as available on the Instances tab.

Created

The date and time the font was created, press the Now button to set it to the current date and time.

Modified

The date and time the font was last modified, press the Now button to set it to the current date and time.

Update modified timestamp on export (recommended)

When checked, the timestamp will be automatically set to the current date and time when you export a font.

Legal

This section contains the **Copyright, Trademark, Licensing, Vendor, and Designer** information.

Embedding Licensing Rights

Embeddable fonts may be stored in a document. When a document with embedded fonts is opened on a system that does not have the font installed (the remote system), the embedded font may be loaded for temporary (and in some cases, permanent) use on that system by an embedding-aware application. Embedding licensing rights are granted by the designer (or vendor) of the font.

Installable (no embedding restrictions)

Fonts that have none of the flags set are installable embedding fonts. Fonts with this setting indicate that they may be embedded and permanently installed on the remote system by an application. The user of the remote system acquires the identical rights, obligations and licenses for that font as the original purchaser of the font, and is subject to the same end-user license agreement, copyright, design patent, and/or trademark as was the original purchaser.

Editable (read-write)

Fonts with this flag set indicate that they may be embedded in documents, but must only be installed temporarily on the remote system. In contrast to Preview & Print fonts, documents containing Editable fonts may be opened “read-write;” editing is permitted, and changes may be saved.

Preview & Print (read-only)

Fonts with this flag set indicate that they may be embedded within documents but must only be installed temporarily on the remote system. Any document which includes a Preview & Print embedded font must be opened "read-only;" the application must not allow the user to edit the document; it can only be viewed and/or printed.

Restricted (embedding is not allowed)

Fonts that have this flag set must not be modified, embedded or exchanged in any manner without first obtaining permission of the legal owner.

No subsetting

When this flag is set, the font may not be subsetted prior to embedding.

Bitmap embedding only

When this flag is set, only bitmaps contained in the font may be embedded. No outline data may be embedded.

Vendor ID and Vendor (Manufacturer)

The Vendor ID identifies the company responsible for the marketing and distribution of the typeface. FontCreator uses information that comes from the vendor id database which is maintained by Microsoft. You can register a four-letter Vendor ID here:

[Register as a font vendor](#)

Designer

The name of the designer of the typeface.

Sample Text

This can be the font name, or any other text that the designer thinks is the best sample to display the font in. Use the Validate button to ensure all characters are available in the font.

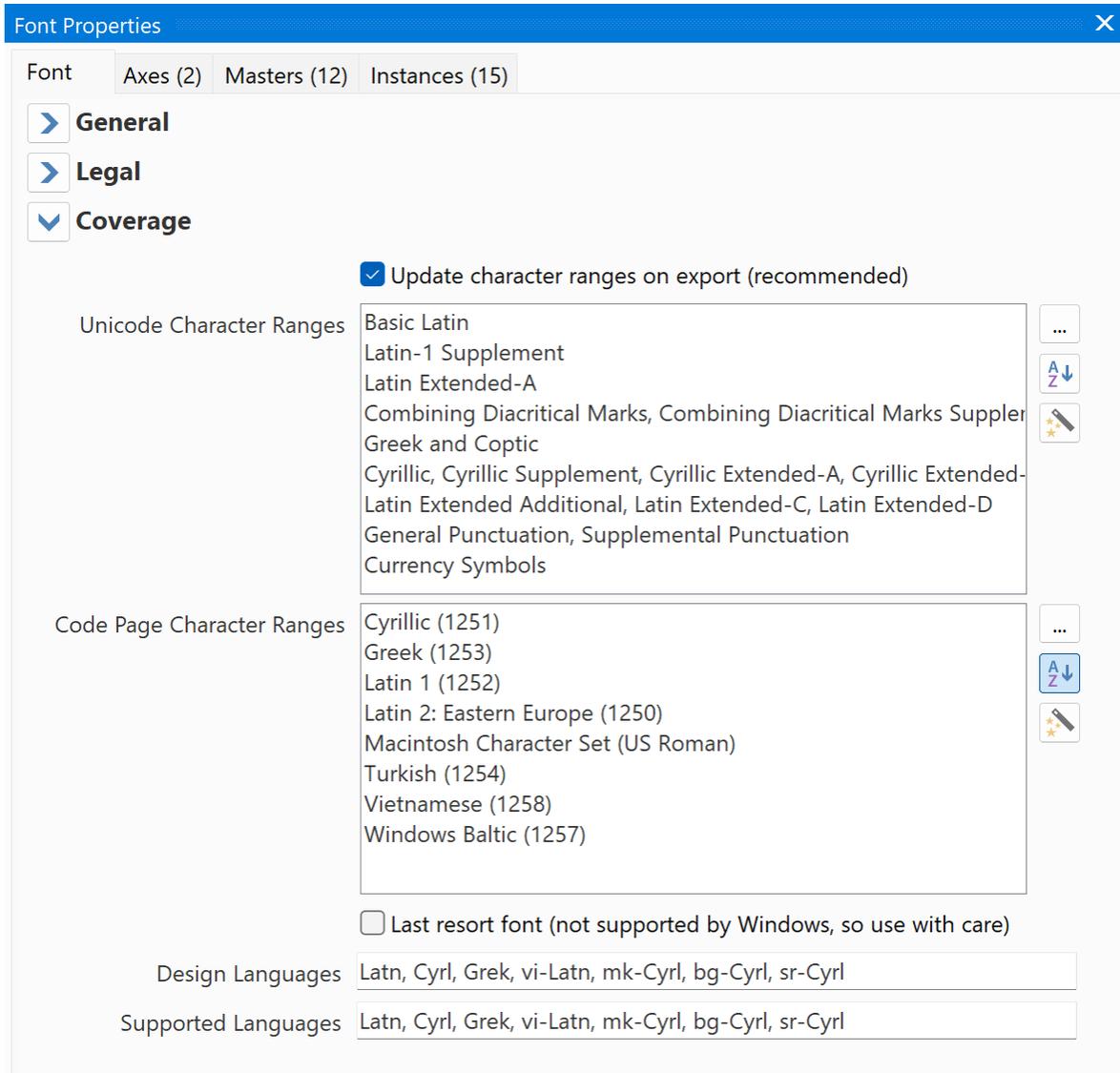
In Windows the sample text is used within the Font Settings preview (Settings -> Personalization -> Fonts). If left empty, Windows (SettingsHandlers_Fonts.dll) will use one of several predefined preview samples:

- Over the horizon comes the break of dawn.
- The aroma of baking bread fills the air.
- Meteors created a sky symphony of light.
- The sound of ocean waves calms my soul.
- Melodic rain bounces off the roof top.
- Your presence might be invaluable.

Variations PostScript Prefix

For variable fonts only and only letters A-Z, a-z, and digits 0-9 are allowed (not even a space or hyphen!). Mainly used to support legacy applications and for construction font names for individual instances when printing. If omitted, they should use Typographic Family Name as prefix. If Typographic Family Name is not provided, then Family Name is considered to be the Typographic Family Name. All characters except letters A-Z, a-z, and digits 0-9 are removed. Thus if Typographic Family Name is **Café Del Sol**, it becomes **CafDelSol**. In such case it is best to provide CafeDelSol as name, to avoid possible ambiguity.

Coverage



Update character ranges on export (recommended)

When enabled the character ranges will be updated when the font is exported. This will make sure that the ranges are always correct.

Unicode Character Ranges

This field is used to specify the Unicode blocks or ranges encompassed by the font file in the mappings for the Windows platform.

If a Unicode range is selected it is considered functional. The determination of “functional” is left up to the font designer, although character set selection should attempt to be functional by ranges if at all possible.

Press the Edit button to modify this field through the Unicode Character Range window or press the Calculate button to generate the value.

Note: All available bits were exhausted as of Unicode 5.1. The bit assignments were last updated for OS/2 version 4 in OpenType 1.5 in 2008. FontCreator uses this (contents and layout) version for the OS/2 table. There are many additional ranges supported in the current version of Unicode that are not supported by these fields in the OS/2 table. See the **Design languages** and **Supported languages** fields below for an alternate mechanism to declare what scripts or languages that a font can support or is designed for.

Code Page Character Ranges

This field is used to specify the code pages encompassed by the font file in the mappings for the Windows platform.

If a code page is selected then the code page is considered functional. The determination of “functional” is left up to the font designer, although character set selection should attempt to be functional by code pages if at all possible.

Press the Edit button to modify this field through the Code Page Character Range window or press the Calculate button to generate the value.

Note: [Legacy symbol fonts](#) (like Webdings and Wingdings) require only Symbol Character Set to be selected. All of the characters in the Unicode range 0xF000 - 0xFFFF (inclusive) will then be used to enumerate the symbol character set. If this code page is not selected, any characters present in that range will not be enumerated as a symbol character set.

Last resort font

If set, indicates that the glyphs are simply generic symbolic representations of the assigned code-point ranges and don't truly represent support for those code-points. If unset, indicates that the glyphs represent proper support for those code-points. If set, FontCreator will use a [compact format](#) for storing ranges of characters mapped to single glyphs.

Warning: FontCreator is able to export last resort fonts, but Windows doesn't support the used format. Right now web browsers and Mac OS X support "true" last resort fonts. If you want to make a last resort font for Windows, then don't set this option.

Design languages

Lists the languages and/or scripts which the font is specifically intended to cover. The list is a series of comma-separated ScriptLangTags. Spaces may be used between tags but are not required. "cy, mi" and "cy,mi" are both valid.

Supported languages

Lists the languages and/or scripts which the font is known to be capable of supporting. Again, this list is a series of comma-separated ScriptLangTags.

To understand the distinction between design and supported languages, consider the case of accented Latin letters. Although the accents are used in common by a number of languages, the precise shape of the accents can depend on the typographic traditions of a specific language. Polish, for example, prefers steeper accents than French. A font which was designed with accents specifically for Polish would then have po-Latn among its 'dIng' tags but Latn among its 'sIng' tags.

ScriptLangTags

A ScriptLangTag denotes a particular script or language associated with a font. They are based on the IETF BCP 47 specification, "Tags for Identifying Languages". See:

<https://www.rfc-editor.org/info/bcp47>

For codes for the representation of scripts, see:

https://en.wikipedia.org/wiki/ISO_15924

And for short codes for language names, see:

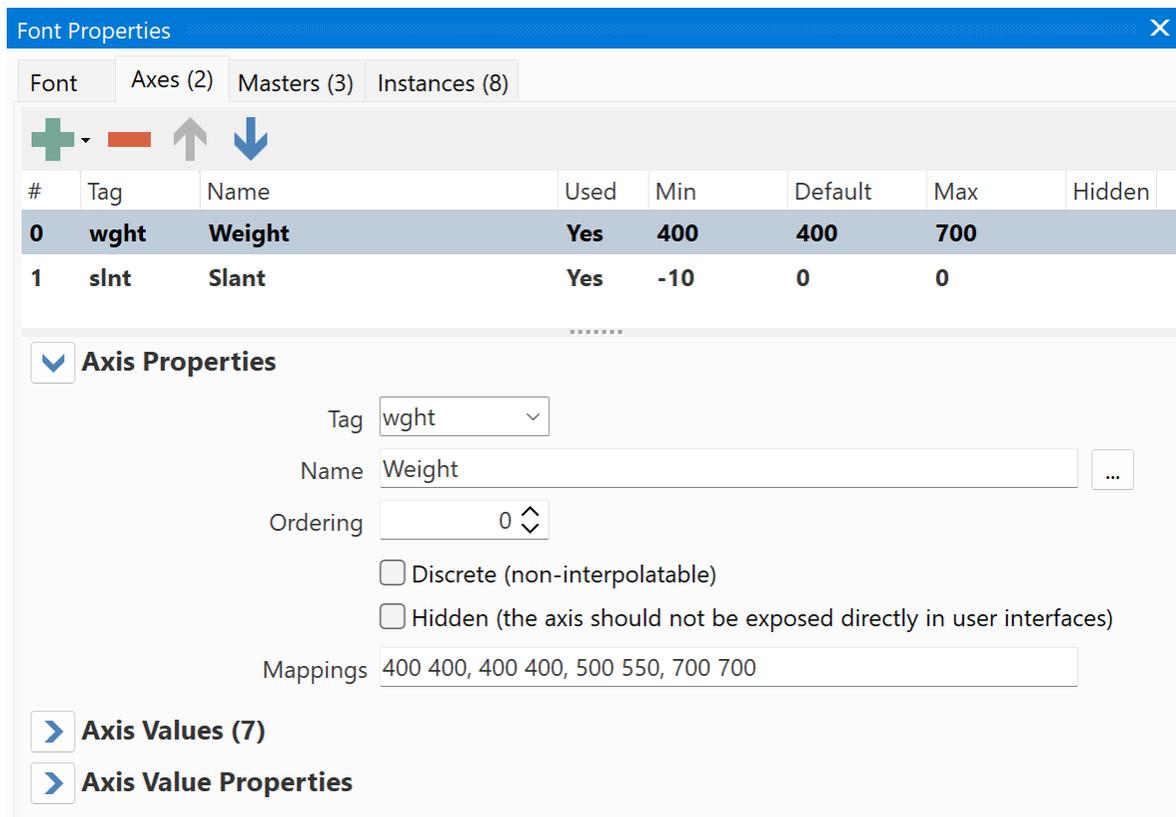
https://en.wikipedia.org/wiki/List_of_ISO_639-2_codes

Thumbnail / Preview/ Font File Icon in Windows File Explorer

With small thumbnails, Windows file explorer shows a standard icon for font files, but with medium icons or larger it displays a couple of characters of the font itself. In general with regular Latin based fonts it shows Abg, but fonts with other scripts may show other characters. What Windows decides to preview depends on the selected Unicode Character Ranges and/or the available Design and Supported Languages. If your font shows math characters and you don't want that, uncheck the "Update character ranges on export" checkbox and manually set the appropriate character ranges.

7.2.2 Axes

The **Axes** tab is only used with variable fonts.



Axes

The most common registered axis is the weight axis, but there are several other registered axes and numerous unofficial axes.

Registered axis tags

Axis tag	Name	Min Value	Max Value	Note
ital	Italic	0	1	A value of 0 can be interpreted as "Roman" (non-italic) and a value of 1 can be interpreted as (fully) italic
opsz	Optical size	> 0		Values can be interpreted as text size, in typographic points
slnt	Slant	> -90	< 90	Values can be interpreted as the angle, in counter-clockwise degrees
wdth	Width	> 0		Values can be interpreted as a percentage of whatever the font designer considers "normal width"
wght	Weight	1	1000	Values can be interpreted in direct comparison to values for Weight Class

To add an axis, click the [+] toolbar button, and select a registered, unofficial, custom or global axis.

A global axis is used among several variable fonts of the same font family. The most common global axis is the Italic axis.

All axis have a minimum, default, and maximum value, except for global axes. These values are **user scale coordinates**, which are derived from axis mappings (see below) and master locations.

An italic font should not include both Italic and Slant axes, unless the font family includes multiple italic designs with different amounts of slant. Most typefaces are made out of two variable fonts. One for the Roman and the other for the Italic styles. In such case both fonts should contain a **global** Italic axis, which is not used for interpolation.

Ordering

A value that applications can use to determine primary sorting of face names, or for ordering of labels when composing family or face names. Numbering starts with 0. To ensure consistency in how face names are presented to users, the axis ordering

should be consistent across different fonts within a family, and with the style names of the instances.

Note: it seems not all software make use of this ordering, so it might be best to reorder the axes through the up and down arrow toolbar buttons.

Discrete (non-interpolatable)

This option is not something that ends up in a font, but it is used to determine whether an axis is part of interpolation or not. It can be used to design a font family as one variable font, while on exporting it is divided into separate variable fonts, so all discrete axes become global.

Hidden (the axis should not be exposed directly in user interfaces)

Indicates a recommendation by the font developer that the axis not be exposed directly to end users in application user interfaces. Reasons for setting this flag might include that the axis is intended only for programmatic interaction, or is intended for font-internal use by the font developer. If this flag is set, the axis should not be exposed to users in application user interfaces except in specialized scenarios, such as a font inspection utility.

Mappings

Optionally provide axis mappings to modify aspects of how a design varies for different instances along a particular axis.

While designing a font, design coordinates are used, but they do not have to represent the actual user scale coordinates. For example, it is common to use the stem width as design values for Weight, but those need to be mapped to actual Weight values for the user. For example a variable font with only a Weight axis, can have a stem width for the Regular weight of 60 font units, while for the end user the actual Weight value will be 400.

A valid collection of map values is one that includes at least a mapping pair for the minimum, default, and maximum values. If default is equal to the minimum or maximum value, then that mapping has to be included twice. Otherwise the mapping is considered invalid and will be ignored. A mapping pair consists of two values, the first value in the user space, and the second value in the design space. For example:

100 16, 200 36, 300 56, 400 72, 500 108, 600 138, 700 170

The conceptual effect of these additional scale mappings is to make the variation along an axis less linear. Values change linearly within each segment, but additional segments make the way that values change across the entire axis range less linear overall. The effect might also be described as compressing some portions of the scale while making other portions less compressed.

Font Properties ✕

Font Axes (1) Masters (3) Instances (4)

+ ▬ ↑ ↓

#	Tag	Name	Used	Min	Default	Max	Hidden
0	wght	Weight	Yes	300	400	700	

> **Axis Properties**

v **Axis Values (4)**

+ ▬ ⚡

Format	Name	Values	Linked	Flags
Value	Light	wght 300		
Value	Regular	wght 400		Elided
Value	SemiBold	wght 600		
Value	Bold	wght 700		

v **Axis Value Properties**

Format

Name ...

Axis

Min Value

Default Value

Max Value

Linked Value

Elidable axis value name (represents the normal value)

Older sibling font attribute

Elided Fallback Name ...

Axis Values

An Axis Value provides details regarding a specific style-attribute value on some specific axis of design variation, or a combination of design-variation axis values, and the relationship of those values to labels used as elements in subfamily names. This information can be useful for presenting fonts in application user interfaces. It is also used by platforms to provide compatibility between rich typographic families with a wide range of styles and older applications that use legacy font family models.

These values are **user scale coordinates**. In a variable font, it is strongly recommended that axis values be included for every element of typographic subfamily names for all of the named instances.

Use the Generate from Instances toolbar option, to generate the axis values based on the Style Names and Axis Locations of all provided Instances.

Format

There are currently four different axis value formats.

1. Value, allows you to associate a specific axis value with a name (e.g. Bold -> 700 for the weight axis)
2. Range, allows a nominal value along with minimum and maximum value for a specific Name. Minimum value supports negative infinity by providing "-INF", and maximum value supports positive infinity by providing "INF".
3. Linked, allows you to associate a specific axis value with a name, along with a style-linked mapping from this value (e.g. Regular -> 400 & Linked -> 700 for the weight axis)
4. Multi-axis, allows two or more axis value combinations.

Elidable axis value name

If set, it indicates that the axis value represents the "normal" value for the axis and may be omitted when composing name strings.

Older sibling font attribute

If set, the axis value provides axis value information that is applicable to other fonts within the same font family. This is used if the other fonts were released earlier and did not include information about values for some axis. If newer versions of the other fonts include the information themselves and are present, then this axis value is ignored.

Elided Fallback Name

A name (such as "Regular") that can be used when composing a name if all of the axis value names are elidable. For example, "Normal" weight and "Roman" slant may both be marked as elidable axis value names, and so a composed name for normal weight and Roman slant may result in an empty string.

7.2.3 Masters

A non-variable font requires one master, while variable fonts need multiple masters as needed for interpolation to work within the variation space.

Masters

Font Properties

Font Axes (3) Masters (12) Instances (15)

P...	Name	wght	width
D	Regular	72	100
	Light	16	100
	Bold	170	100
	Light Condensed	16	75
	Condensed	72	75
	Bold Condensed	170	75

General

Name

Include metrics

Axis Location

Weight

Width

Metrics for Horizontal Writing Mode

Metrics for Vertical Writing Mode

Additional Metrics

Hinting

Note

For a variable font, be sure to include masters with axis locations that allow interpolation. It is common to add masters at the extremes (minimum and maximum axis coordinates), but this is not always necessary. Just remember, a variable font encompasses a whole design space, which is consist of regions which are always rectilinear. A font with two axes, weight and width, can work just fine with these masters:

Master	Weight	Width
--------	--------	-------

Light	300	100
Regular (default)	400	100
Bold	700	100
Condensed	400	75
Ultra-expanded	400	200

So, it might not be needed to include corner masters, like Bold Ultra-condensed (Weight 700, Width 200). And if you later discover there is a need for it, you can add such master later in the design process.

Some glyph outlines need additional layers, e.g. outlines with curves that have inflection points (a change in direction of curvature), like "S". Then add an additional master and only add layers for these outlines. No need to add layers for all glyphs, if they already interpolate the way you want.

Name

The name of the masters is only used while designing the font. It is not part of the exported font. Therefore localisation is not supported here.

Axis Location

Each master has a particular position within the variation space of a variable font. The dimensions of the whole space is defined by the axes. The axis location values are **design scale coordinates**. The user scale coordinates may be different if an axis has mappings.

Metrics for Horizontal Writing Mode

Font Properties ✕

Font Axes (3) Masters (12) Instances (15)

P...	Name	wght	width
D	Regular	72	100
	Light	16	100
	Bold	170	100
	Light Condensed	16	75
	Condensed	72	75
	Bold Condensed	170	75

.....

> General

> Axis Location

▾ Metrics for Horizontal Writing Mode

Typo Ascender 

Typo Descender 

Typo Line Gap 

Win Ascent 

Win Descent 

Ascender 

Descender 

Line Gap 

Cap Height 

x-Height 

Italic Angle (degrees) 

Caret Slope (degrees) 

Caret Offset 

Use typo metrics for line spacing

> Metrics for Vertical Writing Mode

> Additional Metrics

> Hinting

> Note

Calculate

Calculate

Typo Ascender

The typographic ascender for this font. Remember that this is not the same as the **Ascender** value in the **Metrics** tab, which Apple defines in a far different manner.

The suggested usage for **Typo Ascender** is that it be used in conjunction with **units per em** to compute a typographically correct default line spacing. The goal is to free applications from Macintosh or Windows-specific metrics which are constrained by backward compatibility requirements. These new metrics, when combined with the character design widths, will allow applications to lay out documents in a typographically correct and portable fashion.

For CJK (Chinese, Japanese, and Korean) fonts that are intended to be used for vertical writing (in addition to horizontal writing), the required value for **Typo Ascender** is that which describes the top of the design space (also known as em-square). For example, if the design space of the font extends from coordinates 0,-120 to 1000,880 (that is, a 1000x1000 box set 120 design units below the Latin baseline), then the value of **Typo Ascender** must be set to 880. Failing to adhere to these requirements will result in incorrect vertical layout.

Typo Descender

The typographic descender for this font. Remember that this is not the same as the **Descender** value in the **Metrics** tab, which Apple defines in a far different manner.

The suggested usage for **Typo Descender** is that it be used in conjunction with units per em to compute a typographically correct default line spacing. The goal is to free applications from Macintosh or Windows-specific metrics, which are constrained by backward compatibility requirements. These new metrics, when combined with the character design widths, will allow applications to lay out documents in a typographically correct and portable fashion.

For CJK (Chinese, Japanese, and Korean) fonts that are intended to be used for vertical writing (in addition to horizontal writing), the required value for **Typo Descender** is that which describes the bottom of the design space (aka, em-square). For example, if the design space of the font extends from coordinates 0,-120 to

1000,880 (that is, a 1000x1000 box set 120 design units below the Latin baseline), then the value of **Typo Descender** must be set to -120. Failing to adhere to these requirements will result in incorrect vertical layout.

Typo Line Gap

The typographic line gap for this font. Remember that this is not the same as the **Line Gap** value, which Apple defines in a far different manner.

The suggested usage for **Typo Line Gap** is that it be used in conjunction with **units per em** to compute a typographically correct default line spacing. Typical values average 7-10% of units per em.

Win Ascent

The ascender metric for Windows. This, too, is distinct from Apple's Ascender value and from the **Typo Ascender** value. **Win Ascent** is computed as the yMax for all characters in the Windows ANSI character set. Win Ascent is used to compute the Windows font height and default line spacing. For **Symbol fonts**, it is the same as yMax.

Win Descent

The descender metric for Windows. This, too, is distinct from Apple's Descender value and from the **Typo Descender** value. **Win Descent** is computed as the -yMin for all characters in the Windows ANSI character set. **Win Descent** is used to compute the Windows font height and default line spacing. For **Symbol fonts**, it is the same as -yMin.

Ascender (Macintosh-specific)

Typographic ascent

Descender (Macintosh-specific)

Typographic descent

Line Gap (Macintosh-specific)

Typographic line gap. Negative Line Gap values are treated as zero.

Tip: To automatically calculate ascender and descender values press the Calculate button. It will try to preserve original line spacing, so it might also affect line gap.

x-Height

This metric specifies the distance between the baseline and the approximate height of non-ascending lowercase letters measured in Funits. This value would normally be specified by a type designer but in situations where that is not possible, for example when a legacy font is being converted, the value may be set equal to the top of the unscaled and unhinted glyph bounding box of the glyph encoded at U+0078 (LATIN SMALL LETTER X). If no glyph is encoded in this position the field should be set to 0.

This metric, if specified, can be used in font substitution: the xHeight value of one font can be scaled to approximate the apparent size of another.

CapHeight

This metric specifies the distance between the baseline and the approximate height of uppercase letters measured in Funits. This value would normally be specified by a type designer but in situations where that is not possible, for example when a legacy font is being converted, the value may be set equal to the top of the unscaled and unhinted glyph bounding box of the glyph encoded at U+0048 (LATIN CAPITAL LETTER H). If no glyph is encoded in this position the field should be set to 0.

This metric, if specified, can be used in systems that specify type size by capital height measured in millimeters. It can also be used as an alignment metric; the top of a drop capital, for instance, can be aligned to the CapHeight metric of the first line of text.

Italic Angle (degrees)

Italic angle in degrees from the vertical. Zero for upright text, positive for text that leans to the right (forward).

Tip: you can type "x / y" (run/rise) for italic angle and caret slope. Be sure to press Enter to let FontCreator know you intend to set the value as run/rise.

Caret Slope (degrees)

The slope of the caret in degrees from the vertical. The caret is the mark used to indicate where something is to be inserted into a text. It is usually represented by a blinking vertical bar, but with italic (or slanted) fonts it can have an angle.

Tip: you can type "x / y" (run/rise) for italic angle and caret slope. Be sure to press Enter to let FontCreator know you intend to set the value as run/rise.

Caret Offset

The amount by which a slanted highlight on a glyph needs to be shifted to produce the best appearance. Set to 0 for non-slanted fonts. FontCreator uses this value to offset the left side-bearing of glyphs.

Use typo metrics for line spacing

If set, it is strongly recommended to use **Typo Ascender - Typo Descender + Typo Line Gap** as a value for default line spacing for this font.

Metrics for Vertical Writing Mode

Font Properties
✕

Font Axes (3) Masters (12) Instances (15)

+

P...	Name	wght	width
D	Regular	72	100
	Light	16	100
	Bold	170	100
	Light Condensed	16	75
	Condensed	72	75
	Bold Condensed	170	75

General

Axis Location

Metrics for Horizontal Writing Mode

▼ **Metrics for Vertical Writing Mode**

Include (recommended for fonts that are used for vertical writing)

Vertical Typo Ascender

Vertical Descender

Vertical Typo Line Gap

Vertical Caret Slope (degrees)

Vertical Caret Offset

Additional Metrics

Hinting

Note

In general vertical metrics are only useful with fonts that are used for vertical writing. So we recommend to only include them in CJK (Chinese, Japanese, and Korean) fonts. If enabled, it is also useful to show top and bottom bearings. You can enable it within the [Metrics Options dialog](#). Use [Auto Metrics](#) to set vertical spacing of a range of glyphs.

Vertical Typo Ascender

The distance from the ideographic em-box center baseline for the vertical axis to the right of the ideographic em-box and is usually set to $0.5 * \text{units_per_em}$.

Vertical Typo Descender

The distance from the ideographic em-box center baseline for the vertical axis to the left of the ideographic em-box and is usually set to $-0.5 * \text{units_per_em}$.

Vertical Typo LineGap

The vertical typographic gap. An application can determine the recommended line spacing for single spaced vertical text for an OpenType font by the following expression: ideographic em-box width + Vertical Typo LineGap.

Vertical Caret Slope

The slope of the caret in degrees. Set value equal to 90 (a horizontal caret) for regular/nonslanted fonts.

Tip: you can type "x / y" (run/rise) for italic angle and caret slope. Be sure to press Enter to let FontCreator know you intend to set the value as run/rise.

Vertical Caret Offset

The amount by which the highlight on a slanted glyph needs to be shifted away from the glyph to produce the best appearance. Set value equal to 0 for regular/nonslanted fonts.

Additional Metrics

Font Properties
✕

Font
Axes (3)
Masters (12)
Instances (15)

+

P...	Name	wght	width
D	Regular	72	100
	Light	16	100
	Bold	170	100
	Light Condensed	16	75
	Condensed	72	75
	Bold Condensed	170	75

> General

> Axis Location

> Metrics for Horizontal Writing Mode

> Metrics for Vertical Writing Mode

▾ Additional Metrics

Update additional metrics on export

Subscript horizontal font size

Subscript vertical font size

Subscript x offset

Subscript y offset

Superscript horizontal font size

Superscript vertical font size

Superscript x offset

Superscript y offset

Strikeout Position

Strikeout Thickness

Underline Position

Underline Thickness

Calculate

> Hinting

> Note

Note: Word processors that support OpenType layout features may use the Subscript (subs) and Superscript (sup) features, if available in the font.

Subscript horizontal font size

The recommended horizontal size in font design units for subscripts for this font.

Subscript vertical font size

The recommended vertical size in font design units for subscripts for this font.

Subscript x offset

The recommended horizontal offset in font design units for subscripts for this font.

Subscript y offset

The recommended vertical offset in font design units from the baseline for subscripts for this font.

Superscript horizontal font size

The recommended horizontal size in font design units for superscripts for this font.

Superscript vertical font size

The recommended vertical size in font design units for superscripts for this font.

Superscript x offset

The recommended horizontal offset in font design units for superscripts for this font.

Superscript y offset

The recommended vertical offset in font design units from the baseline for superscripts for this font.

Strikeout position

The position of the strikeout stroke relative to the baseline in font design units, this positive values represent distances above the baseline. Aligning the strikeout position with the em dash is suggested. Note, however, that the strikeout position should not interfere with the recognition of standard characters, and therefore

should not line up with crossbars in the font. For a Roman font with a 2048 em square, 460 is suggested.

Strikeout Thickness

Width of the strikeout stroke in font design units. This field should normally be the width of the em dash, and should also match the underline thickness. For a Roman font with a 2048 em square, 102 is suggested.

Underline Position

This is the suggested distance of the top of the underline from the baseline (negative values indicate below baseline).

Underline Thickness

Suggested value for the underline thickness. In general, the underline thickness should match the thickness of the underscore character ($\$5F$), and should also match the strikeout thickness.

Calculate

The calculate button calculates the values so that they look correct in Microsoft Word.

Note: It is up to word-processing software to use these values. However, be aware that not all software uses these values uniformly, so when superscripts, subscripts, strikethrough, and underline look correct in one application, they might look wrong in another application.

Font Properties ✕

Font Axes (3) Masters (12) Instances (15)

P...	Name	wght	wdth
D	Regular	72	100
	Light	16	100
	Bold	170	100
	Light Condensed	16	75
	Condensed	72	75
	Bold Condensed	170	75

General
 Axis Location
 Metrics for Horizontal Writing Mode
 Metrics for Vertical Writing Mode
 Additional Metrics
 Hinting

Preferred rasterization techniques for fonts with TrueType based outlines.

Limit	Standard	ClearType
9	Grayscale	Smoothing
65535	Gridfitting & Grayscale	Gridfitting & Smoothing

Note

Hinting

This describes the preferred rasterization techniques for the typeface when it is rendered on grayscale-capable devices. It also has some use for monochrome devices, which may use the table to turn off hinting at very large or small sizes, to improve performance. It was invented for screen output and will not be used by printer drivers.

Note: smoothing is optional and only works with TrueType based outlines, so it will not be included if you export fonts with CFF based outlines.

If there are no ranges defined in a typeface, the rasterizer may apply default rules to decide how to render the glyphs on grayscale devices. The rasterizer will use the ClearType related values, if ClearType is enabled.

At very small sizes, the best appearance on grayscale devices can usually be achieved by rendering the glyphs in grayscale without gridfitting. At intermediate sizes, gridfitting (also known as hinting) and monochrome rendering will usually produce the best appearance. At large sizes, the combination of gridfitting and grayscale rendering will typically produce the best appearance. Click the **Generate** button to get smoothing settings based on these recommendations.

To add a new range, press the **Add** button. To remove a range, select it and press the **Delete** button.

Press the **Default** button the add common used values.

Note

A note may be useful for you as font designer, but it is not part of the exported font.

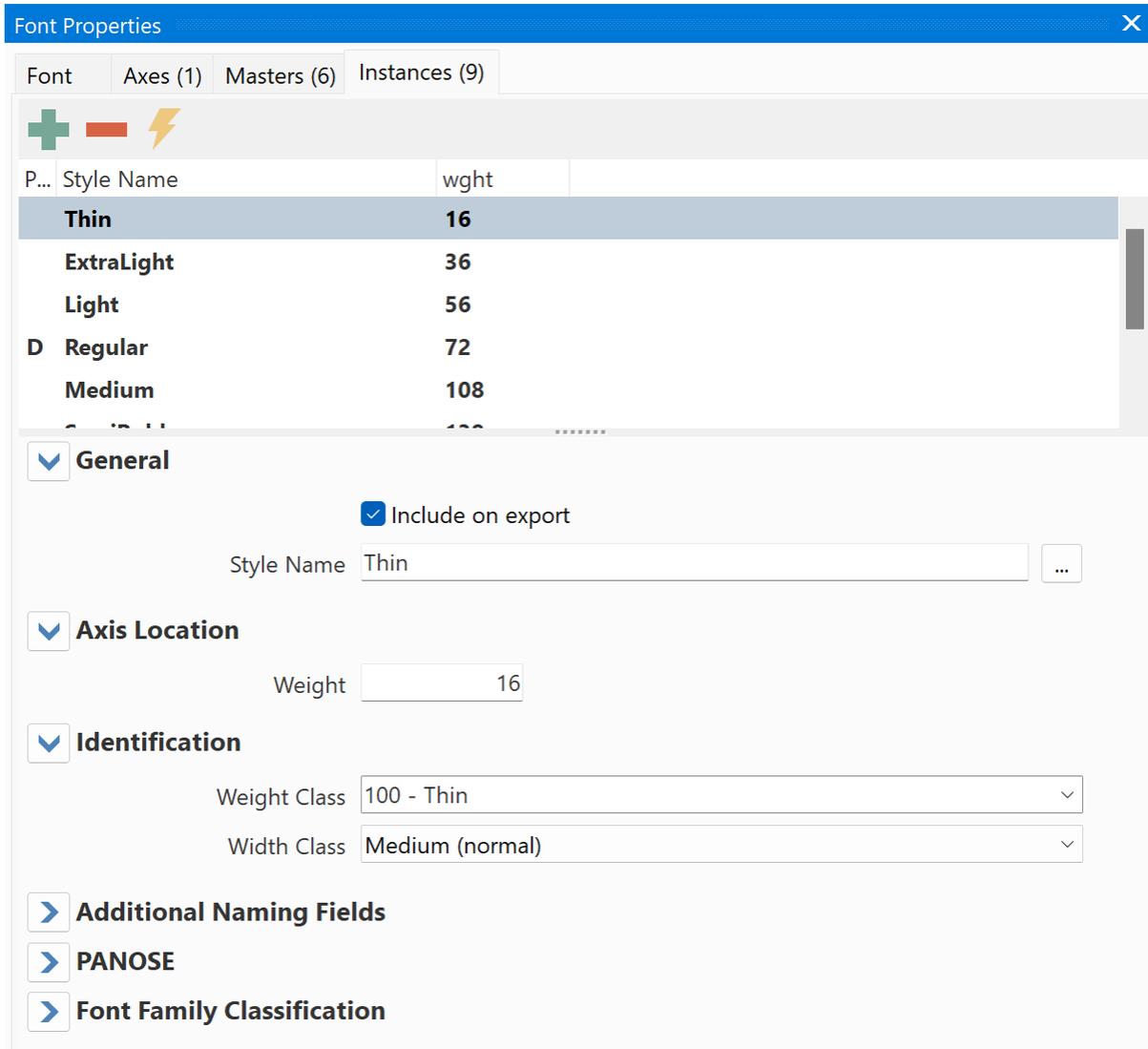
See also:

[Metrics Options](#)

7.2.4 Instances

A non-variable font requires one named instance, while variable fonts most likely end up with several named instances.

Instances



The screenshot shows the 'Font Properties' dialog box with the 'Instances' tab selected. The dialog has a blue title bar with a close button. Below the title bar are tabs for 'Font', 'Axes (1)', 'Masters (6)', and 'Instances (9)'. The 'Instances' tab contains a table of font instances and several configuration sections.

P...	Style Name	wght
	Thin	16
	ExtraLight	36
	Light	56
D	Regular	72
	Medium	108

General

Include on export

Style Name ...

Axis Location

Weight

Identification

Weight Class ▾

Width Class ▾

Additional Naming Fields

PANOSE

Font Family Classification

General

Include on export

Only instances that have this box checked will end up in the exported font. While designing your font, it might be convenient to have several additional instances that help you test interpolation.

Style Name

When the name of a font is exposed in a user interface, it usually is a combination of Family Name (e.g. "My Font") as available from the Font tab, and this Style Name (e.g. "Extra Bold").

Note: The whole set of included instances should be in sync with the Axis Values as defined on the Axes tab. If you have provided all axis values, then the lightning toolbar icon will generate all corresponding named instances.

Axis Location

Each master has a particular position within the variation space of a variable font. The dimensions of the whole space is defined by the axes. The axis location values are **design scale coordinates**. The user scale coordinates may be different if an axis has [mappings](#).

Identification

Weight Class

Indicates the visual weight (degree of blackness or thickness of strokes) of the characters in the font. Minimum value is 1 and maximum value is 1000.

Note: Some (older) versions of Windows and Word automatically add fake bold to any font with weight set lower than 250, e.g. Thin or Extra-light (Ultra-light), so better avoid these weights.

Width Class

Indicates a relative change from the normal aspect ratio (width to height ratio) as specified by a font designer for the glyphs in a font.

Note: Although every character in a font may have a different numeric aspect ratio, each character in a font of normal width has a relative aspect ratio of one. When a new type style is created of a different width class (either by a font designer or by some automated means) the relative aspect ratio of the characters in the new font is some percentage greater or less than those same characters in the normal font -- it is this difference that this parameter specifies.

Relation with Weight and Width Axes

If this is a variable font that has a Weight axis, then both Weight Class value and Axis Value (in user coordinates) should be equal.

If this is a variable font that has a Width axis, then the values should match as follows:

Width Class	Width Axis (in user coordinates)
Ultra-condensed	50
Extra-condensed	62.5
Condensed	75
Semi-condensed	87.5
Medium (normal)	100
Semi-expanded	112.5
Expanded	125
Extra-expanded	150
Ultra-expanded	200

Additional Naming Fields

Generate the following 4 naming fields

In general it is best to let FontCreator decide if and what values to use on exporting the font.

Typographic Family Name & Typographic Subfamily Name (Windows only)

In Windows, the Family name is displayed in the font menu; the Subfamily name is presented as the Style name. For historical reasons, font families have contained a maximum of four styles (regular, italic, bold, and bold italic), but font designers may group more than four fonts to a single family. The Typographic Family and Typographic Subfamily IDs allow font designers to include the preferred

family/subfamily groupings. These IDs are only present if they are different from fields Font Family name and Font Subfamily name. For more information see [Font Family \(typeface\) Settings and Flags](#).

Style Map Family Name & Style Map Style Name

In the official specification, the Style Map Family Name is known as Font Family name (Name ID 1) and Style Map Style Name as Font Subfamily name (Name ID 2). These names are mostly superseded with the current Family Name, as available from the Font tab and Style Name, as available from the Instances tab. The Style Map Family Name is used in combination with Style Map Style Name, and should be shared among at most four fonts that differ only in weight or style.

Generate the following 8 naming fields

In general it is best to let FontCreator decide if and what values to use on exporting the font.

Full Font Name

The full font name reflects all family and relevant subfamily descriptors.

Version String

The version string should begin with the syntax "Version <number>.<number>" (upper case, lower case, or mixed, with a space between "Version" and the number). The string must contain a version number of the following form: one or more digits (0-9) of value less than 65,535, followed by a period, followed by one or more digits of value less than 65,535. Any character other than a digit will terminate the minor number. A character such as ";" is helpful to separate different pieces of version information.

Unique Font Identifier

A unique identifier that applications can store to identify the font being used.

PostScript Name

The PostScript name for the font specifies a string which is used to invoke a PostScript language font that corresponds to this OpenType font. When translated to ASCII, the name string must be no longer than 63 characters and restricted to the printable ASCII subset, codes 33 to 126, except for the 10 characters '[', ']', '(', ')', '{', '}', '<', '>', '/', '%'.

WWS Family Name

The WWS Family name *

WWS Subfamily Name

The WWS Subfamily name *

PostScript CID Findfont Name

Its presence in a font means that the PostScript name field in the Naming window holds a PostScript font name that is meant to be used with the "composefont" invocation to invoke the font in a PostScript interpreter.

This field must be restricted to the printable ASCII subset, codes 33 through 126, except for these 10 characters: [], (), { }, < >, / and %.

Compatible Full (Macintosh only)

On the Macintosh, the menu name is constructed using the FOND resource. This usually matches the Full Font Name. If you want the name of the font to appear differently than the Full Font Name, you can insert the Compatible Full name in this field.

*) For more information about WWS please visit our forums at <https://forum.high-logic.com/> or visit this website:
<https://blogs.msdn.microsoft.com/text/2007/04/23/wpf-font-selection-model/>

Font Properties ✕

Font Axes (3) Masters (12) Instances (15)

P...	Style Name	wght	wdth
	Light	300	100
	SemiLight	350	100
	Regular	400	100
	SemiBold	549.99542	100
	Bold	700	100

General
 Axis Location
 Identification
 Additional Naming Fields
 PANOSE

Value Show hexadecimal

Family Kind

Serif Style

Weight

Proportion

Contrast

Stroke Variation

Arm Style

Letterform

Midline

X-height

Note: for monospaced fonts set Family Kind to 2 and Proportion to 9

Font Family Classification

Class

Subclass

PANOSE

These fields are used to describe the visual characteristics of a given typeface. These characteristics are then used to associate the font with other fonts of similar appearance having different names. The PANOSE evaluation document details the specifications for assigning PANOSE numbers.

<http://www.panose.com/> 

Font Family Classification

These fields are a classification of the font-family design.

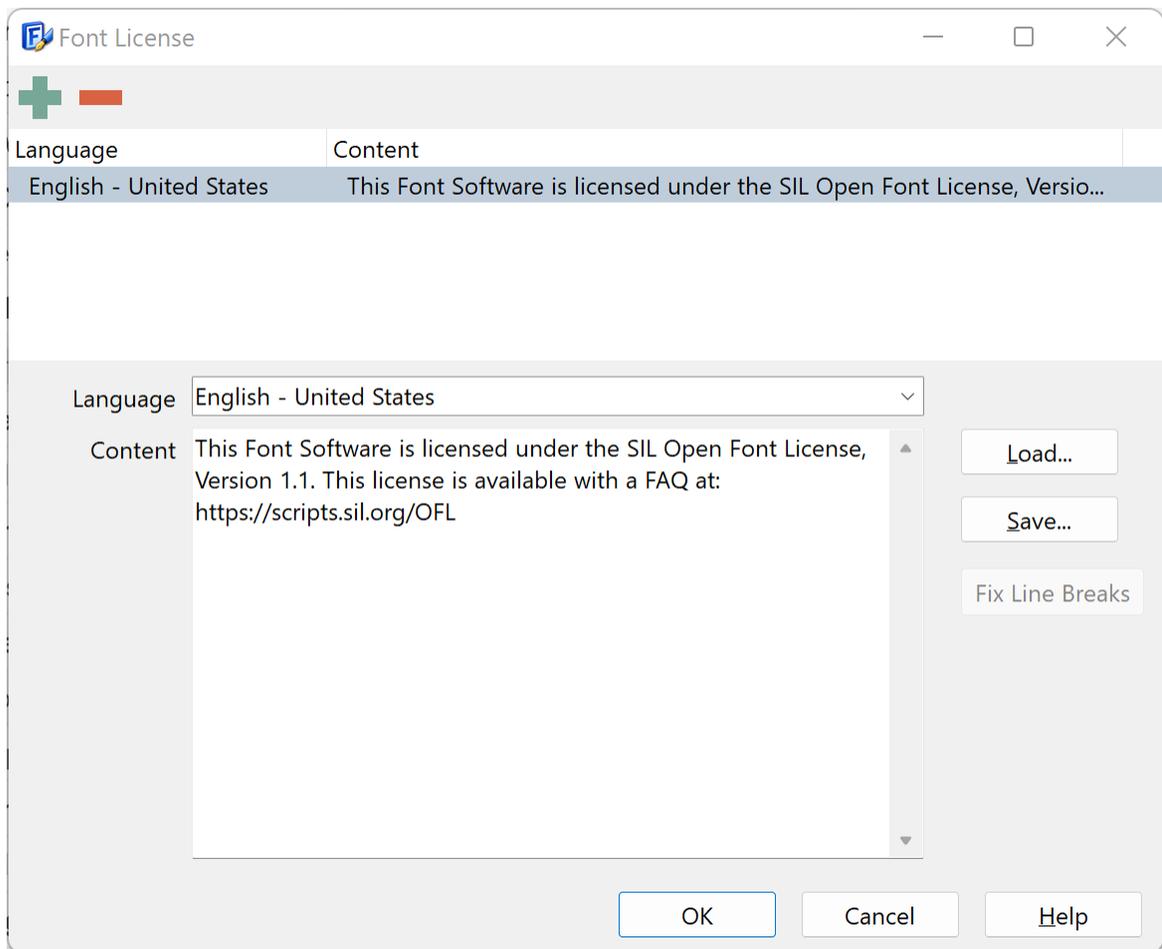
The font class and font subclass are registered values assigned by IBM to each font family. These fields are intended for use in selecting an alternate font when the requested font is not available. The font class is the most general and the font subclass is the most specific. More information about this field is available online:

<https://docs.microsoft.com/en-us/typography/opentype/spec/ibmfrc> 

7.2.5 Naming Fields

Several naming fields can be modified within the Edit Naming Field window, accessed by pressing the [...] button.

The dialog allows you to edit a naming field, and also lets you add language specific **naming** entries, for example for CJK fonts to specify Chinese, Japanese, or Korean naming fields.



You can include the current year through this variable: <Year>

These variables can be used within custom naming fields, and will be replaced by the specific default (English - United States) naming field:

- <Family>
- <Style>
- <Copyright>
- <Trademark>
- <License Agreement>
- <License Agreement URL>

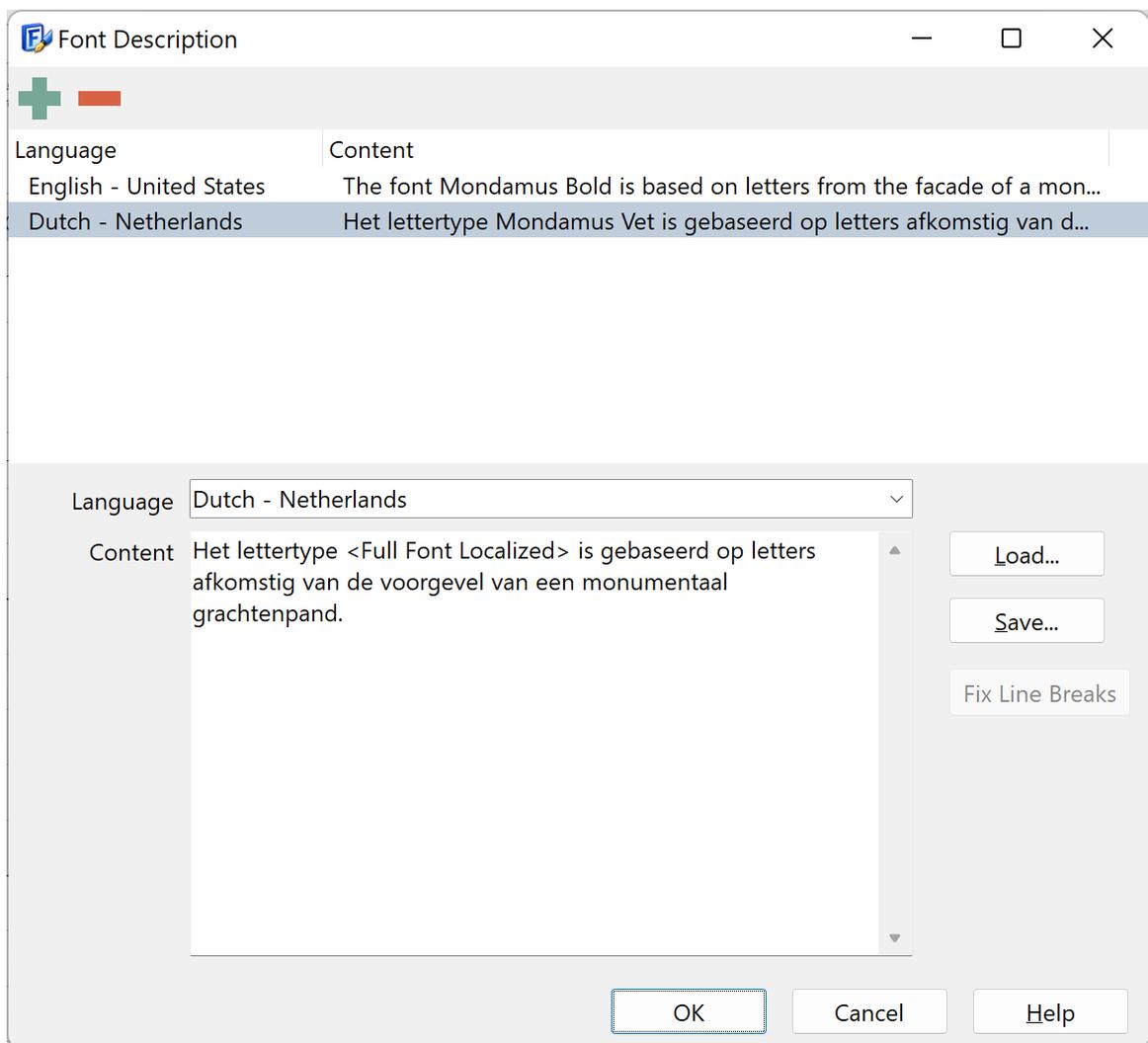
- <Vendor>
- <Vendor URL>
- <Designer>
- <Designer URL>
- <Typographic Family>
- <Typographic Subfamily>
- <Style Map Family>
- <Style Map Subfamily>
- <Full Font>
- <Version>
- <Unique Font ID>
- <PostScript>

There are some more localized variables, which will be replaced by the localized naming field:

- <Family Localized>
- <Style Localized>
- <Full Font Localized>

For example if you want to include the Dutch full font name in the Dutch description, you can use:

Het lettertype <Full Font Localized> is gebaseerd op letters afkomstig van de voorgevel van een monumentaal grachtenpand.



The following are deprecated, and will no longer work:

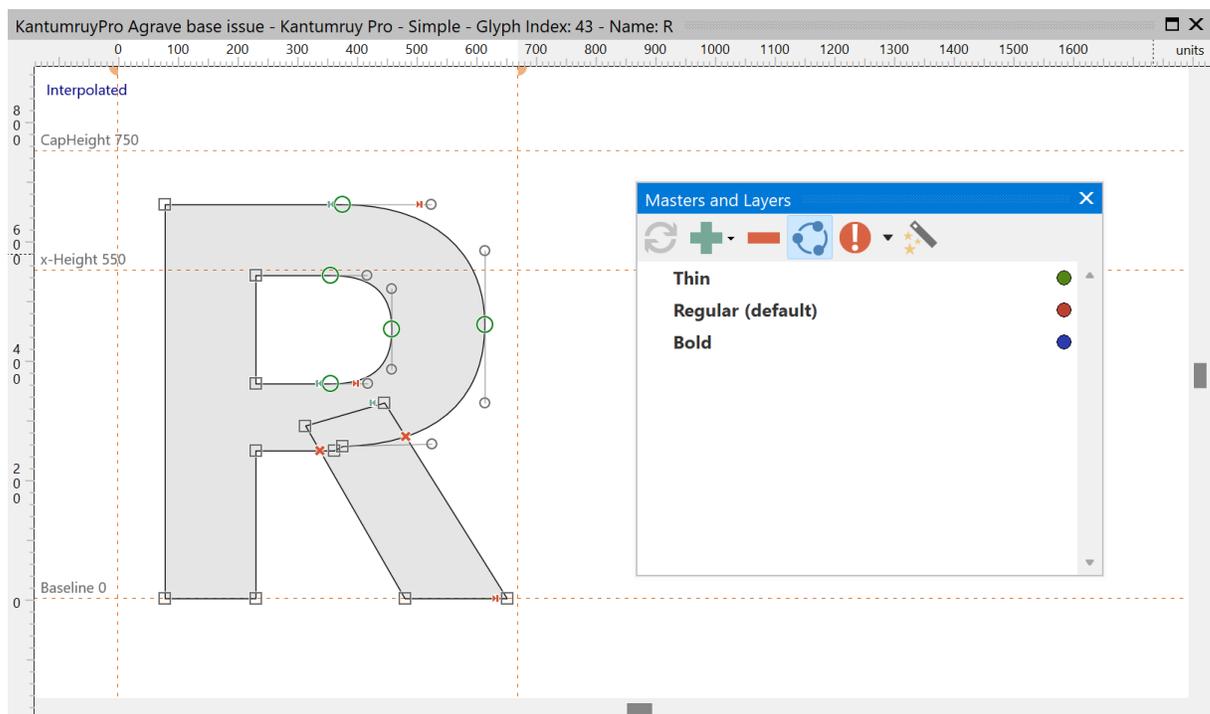
- -> <Family Name>
- -> <Style Name>
- -> <Family Localized>
- -> <Style Localized>
- <Localized>

Fix Line Breaks will be enabled if the text contains incorrect line breaks; most likely because the original text was generated in an Unix environment.

7.3 Masters and Layers

The Masters and Layers panel shows all masters that are used by either the font panel or a glyph panel. This panel allows you to add multiple masters to a variable font.

In order to support interpolation, a variable font requires at least two masters with compatible glyph outlines. With variable fonts, overlaps are explicitly allowed, as shown below.



Editing Across Layers

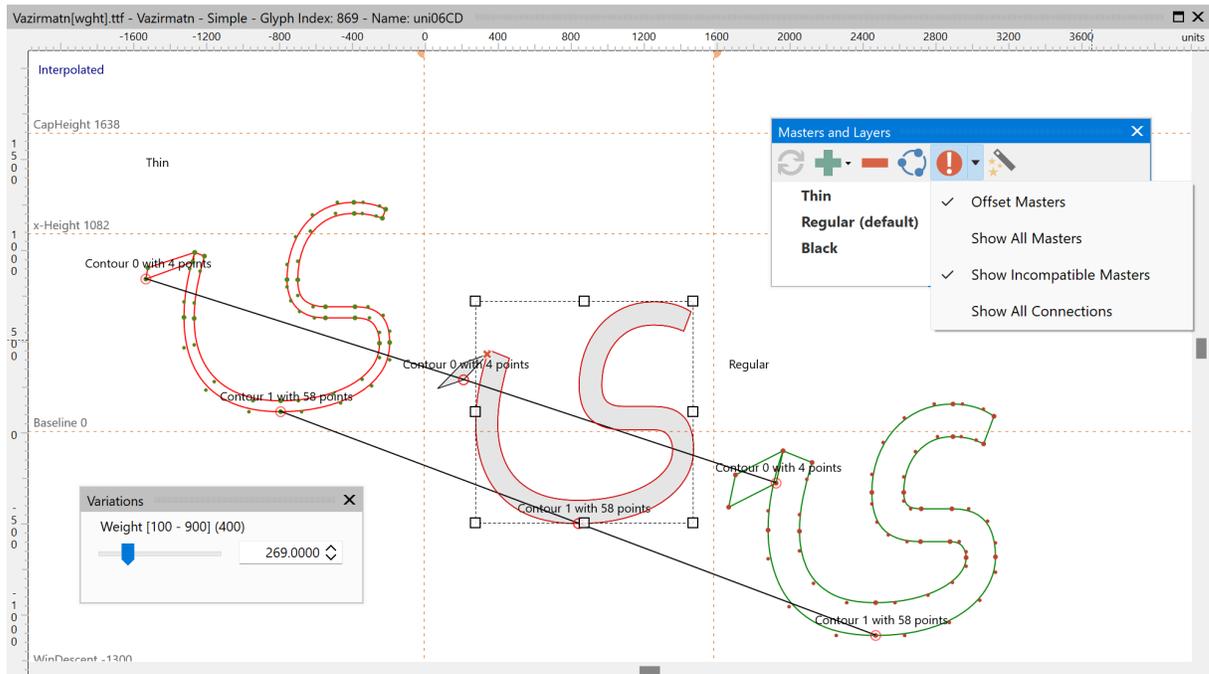
The Masters and Layers panel has a "Edit Across Layers" toolbar button, which when enabled will perform several operations, like delete points, for all glyph layers at once. For most of such actions, it is required that layers are compatible. If you only want to edit the active layer, then make sure the "Edit Across Layers" option is not enabled.

Compatibility and Interpolation Issues

It has toolbar icons that help identify compatibility issues and even to fix interpolation issues.

FontCreator but can locate and fix these issues:

- Mixed up contours and/or glyph members
- Start points not matching
- Different contour directions



The Masters and Layers panel will show a red icon for each incompatible master, followed by one or more letters that relate to interpolation issues:

O -> Outlines

M -> OutlineContourMismatch or OutlineComponentMismatch

F -> OutlineContourFlavour (layers contain mixed quadratic and cubic contours)

N -> OutlineNodeType (mismatch in On and Off curve nodes)

D -> OutlineContourDirection

S -> OutlineContourStart

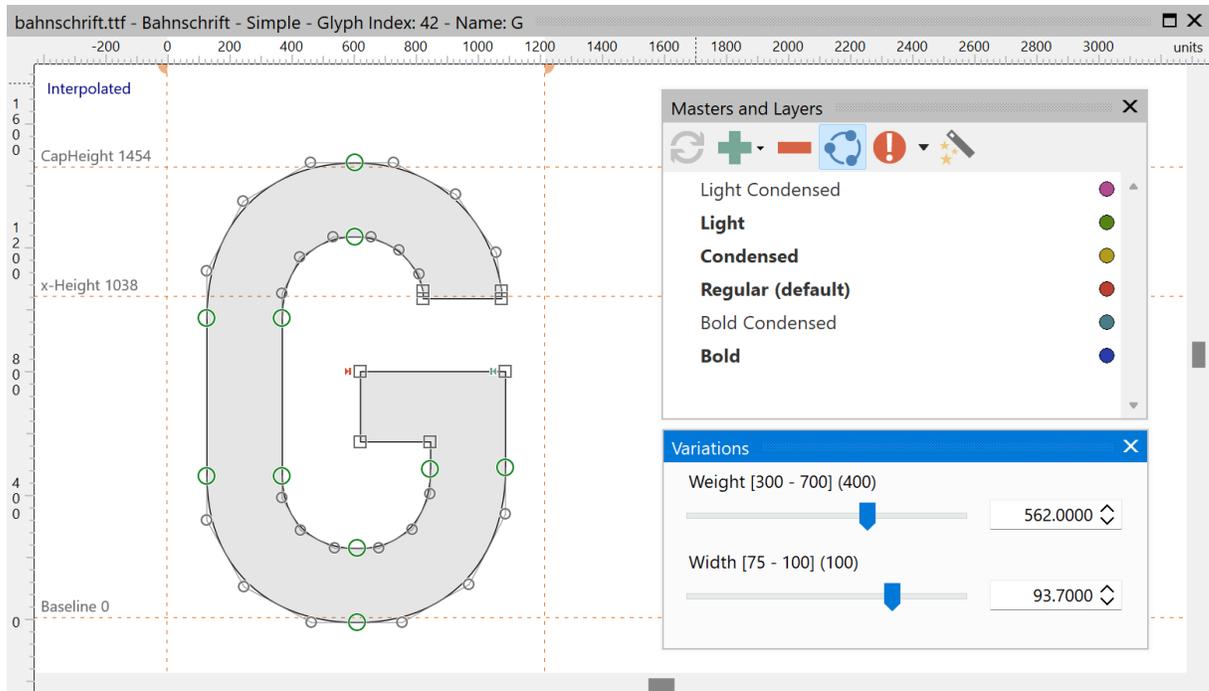
W -> OutlineContourSwapped or OutlineComponentSwapped

T -> OutlineGlyphMatrix (Scaling, Rotation, Skewing)

A -> Anchors

7.4 Variations

The Variations panel is only used with variable fonts. It allows you to change axis coordinate values, so you can preview interpolated glyph outlines.

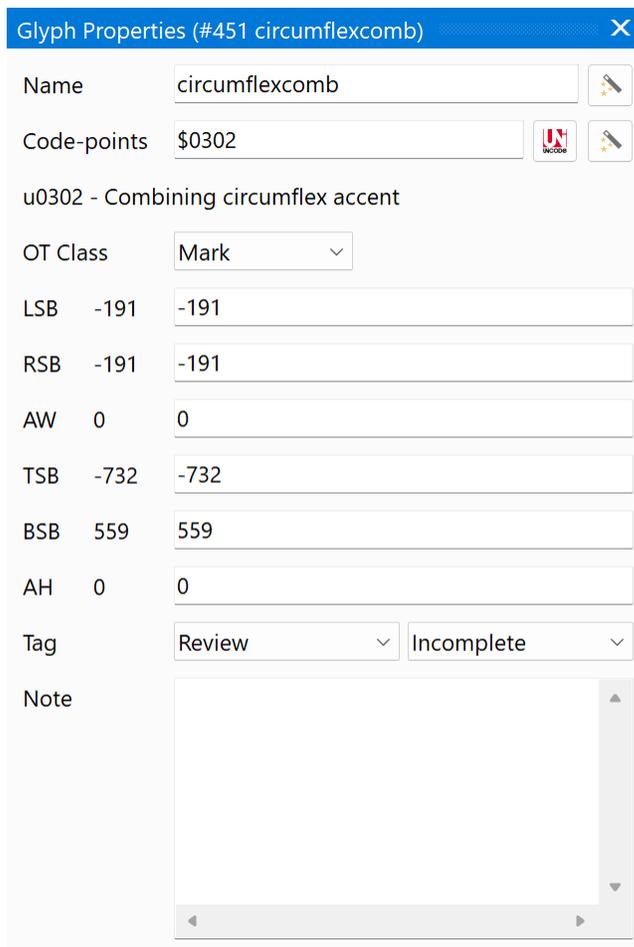


The axis location values are design scale coordinates.

7.5 Glyph Properties

7.5.1 Glyph Properties

The **Glyph Properties** panel provides a quick way to view and edit some of the most common glyph properties.



The screenshot shows the 'Glyph Properties' dialog box for the glyph '#451 circumflexcomb'. The dialog has a blue title bar with a close button. It contains several fields and controls:

- Name:** A text field containing 'circumflexcomb' and a 'Generate Name' button.
- Code-points:** A text field containing '\$0302', a 'Select Unicode Character' button, and a 'Generate code-point' button.
- u0302 - Combining circumflex accent:** The glyph name.
- OT Class:** A dropdown menu set to 'Mark'.
- LSB:** A field with '-191' on the left and a text input field containing '-191'.
- RSB:** A field with '-191' on the left and a text input field containing '-191'.
- AW:** A field with '0' on the left and a text input field containing '0'.
- TSB:** A field with '-732' on the left and a text input field containing '-732'.
- BSB:** A field with '559' on the left and a text input field containing '559'.
- AH:** A field with '0' on the left and a text input field containing '0'.
- Tag:** Two dropdown menus, the first set to 'Review' and the second set to 'Incomplete'.
- Note:** A large empty text area with a scrollbar.

Name

The name of the glyph. Press the Generate Name button to let FontCreator fill in the field.

Code-points

The code-points assigned to this glyph. Press the **Select Unicode Character** button to select a character from the Unicode Character list. Press the **Generate code-point** button to let FontCreator fill in the field automatically. It is possible to enter multiple code-points by separating them with commas, but usually only one character is mapped to each glyph.

Note: You can switch between decimal and hexadecimal values for the code-points in the View tab of the Options dialog.

Unicode name

The Unicode name of the first code-point.

OT Class

The OpenType Class, which is important for the [OpenType Layout Features](#), and is therefore also used within the OpenType Designer window to specify if a glyph should be ignored (Ignore Base Glyphs, Ignore Ligatures, etc.) when a lookup is processed by a layout engine. It can be set to Automatic (so FontCreator determines the actual type), Unassigned, Base, Ligature, Mark, or Component. With Ligatures, an additional edit box allows you to specify the number of components. This defines the number of anchor values used with [MarkToLigature lookups](#).

Note: You can set OT Class to Component, but in general it is not useful. We provide the option, as it is part of the OpenType specification.

Expressions used with Glyph Metrics (LSB, RSB, AW, TSB, BSB, AH)

The glyph metric fields do support expressions. More information about them can be found here: [Glyph Metrics](#)

LSB (Left Side Bearing)

The (horizontal) start position of the glyph.

Note: when you change the left side bearing, all anchors will move along with the glyph outline.

RSB (Right Side Bearing)

The (horizontal) end position of the glyph.

AW (Advance Width)

The width of the glyph and its surrounding space.

Tip: You can also adjust the Left Side Bearing and Advance Width in the Glyph panel, by dragging the vertical bearings.

TSB, BSB, and AH

These fields refer to vertical metrics, namely top side-bearing, bottom side-bearing, and advance height. These values are only available if vertical spacing is enabled. You can enable vertical spacing at the Font Properties panel on the Masters tab.

Tag (Glyph Tag and Layer Tag)

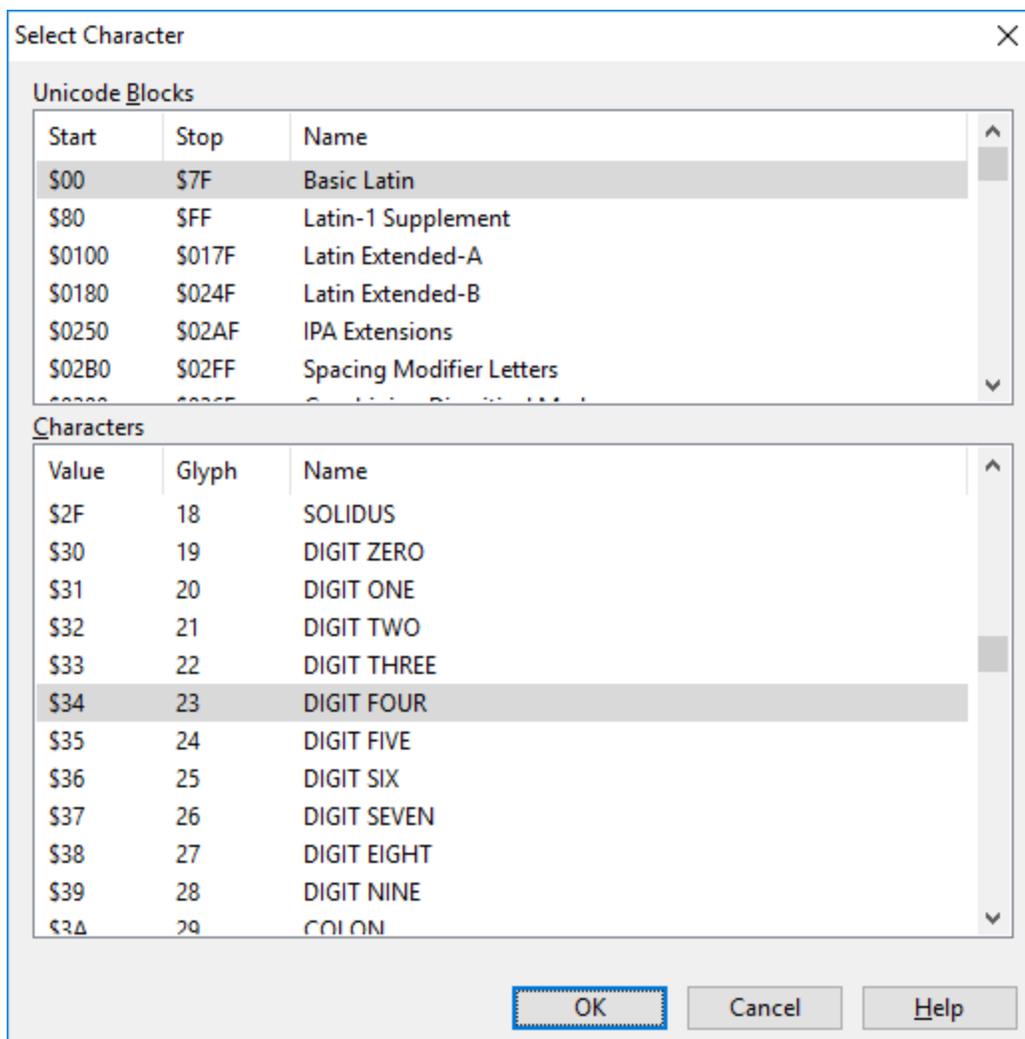
Tags allow you to mark glyphs so they appear with a background color in the Font panels. See [Tags](#) for more information.

Note

Optionally provide a note for your own reference. Notes will not end up in the exported font.

7.5.2 Select Character

The **Select Character** dialog provides a quick way to change a character mapping. This dialog is available through the **Glyph Properties** panel by clicking the **Select Character** button .



7.6 Transform

The Transform panel contains powerful tools for editing glyphs. It can be toggled on and off using the F6 shortcut key, but can only be used while editing glyph outlines in the Glyph panel.

Both contours and components can be precisely repositioned, resized, scaled, rotated, or skewed. Nodes can be repositioned by precise increments.

Transform

X 288

Y 32

W 960 H 1216

100 % 100 %

0 ° 0 °

Lock aspect ratio

Move selection

You can use expressions within the X and Y position fields, which allows to make movement. For example, if a component is positioned at X = 50 and you want to move it by 150 units, then provide this text into the edit box: 50+150.

Lock aspect ratio

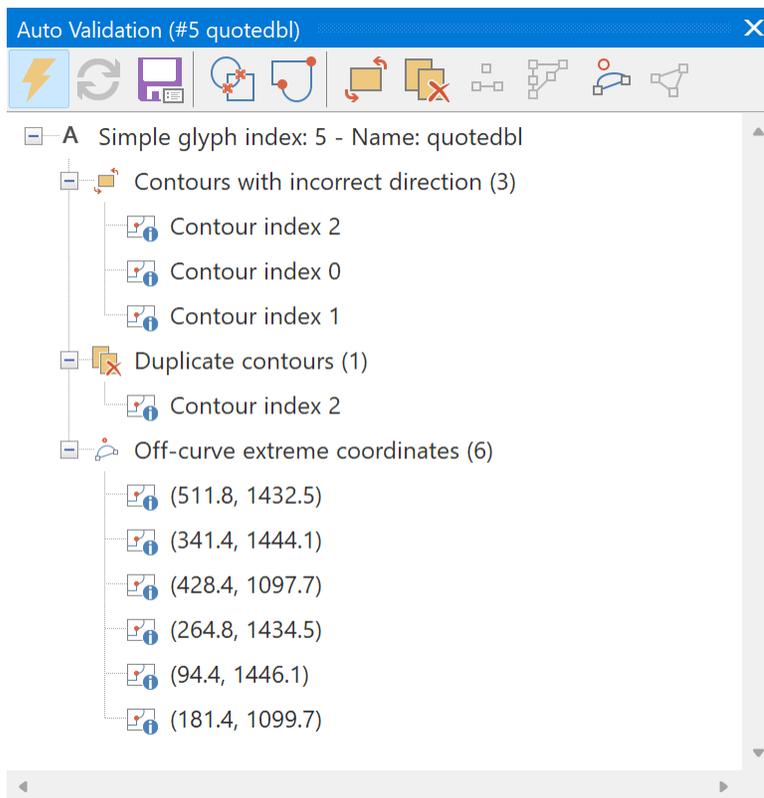
If checked, it preserves the relative width and height or horizontal and vertical scale.

Note: prior to version 12.0.0.2535 you could also set the glyph's side-bearings. This is now solely available through the [Glyph Properties panel](#).

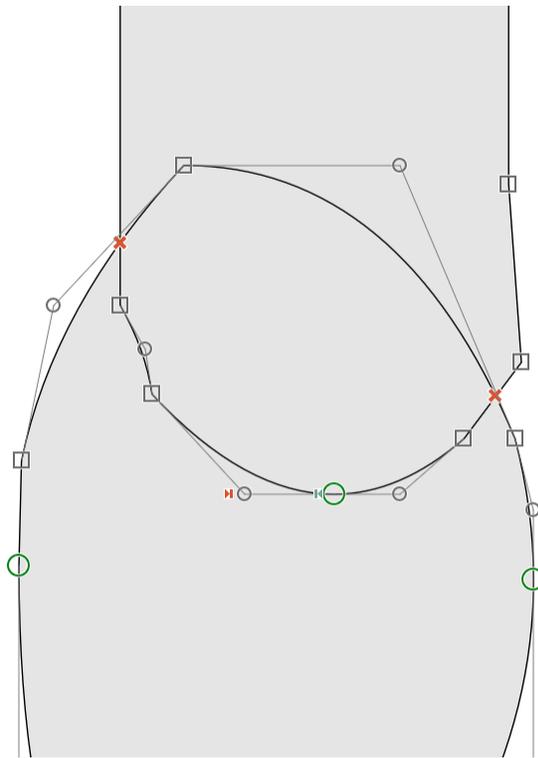
7.7 Validation

Use the **Validation** panel to locate and solve common glyph problems. It can be toggled on and off using the **F7** shortcut key (or **Show glyph validation report** button on the **Glyph** toolbar), but is only available in the **Glyph** panel.

Note: With variable fonts, overlaps are explicitly allowed, so intersecting coordinates are not an issue.



Note: Because validation can be very time-consuming, real-time glyph problem validation will be disabled for too complex glyphs. The limits for real-time glyph validation are set on the **Validation** tab of the **Options** dialog.



Red marks in the **Glyph** panel will show the position of the located problems. Not every problem should be classified as an error, it is the designer's decision to correct or ignore potential problems. Double-click on a reported problem to focus the problematic point or contour. There are several buttons on the **Validation** panel to perform actions that will automatically solve some specific problems.

Enable real-time glyph problem validation

Use this button to enable and disable real-time glyph problem validation.

Refresh

When real-time validation is not enabled, use this button to revalidate the glyph.

Save report

Use this button to save the report to a file.

Show intersecting components and contours

Intersections will be shown when this button is enabled and is down. Red crosses in the **Glyph** panel will show the position of the intersections.

Show warning points

Warning points will be shown when this button is enabled and is down. Red bullets in the **Glyph** panel will show the position of the located problems.

Correct contour directions

Use this button to correct the direction of all misoriented contours in a simple glyph. This button is only enabled when contour direction problems are detected.

Note: This test will not be performed when **Duplicate contours** or **Intersecting coordinates** have been reported.

Remove duplicate components and contours

This button will remove duplicate components from a composite glyph and will remove duplicate contours from a simple glyph.

Remove empty components and contours with one and two points

This button will remove empty components from a composite glyph and will remove contours with one and two points from a simple glyph.

Remove redundant points

Press this button to remove all redundant points.

Note: This feature won't remove duplicate knots as this would affect the outline.

Add on-curve extremes

Press this button to add on-curve extremes. This feature will add global or local extremes, as customized through the validation tab of the [Options dialog](#).

Note: As this feature generates new points, this could lead to new redundant points.

Note: The **Validation** features are not available in the Home Edition of FontCreator.

See also:

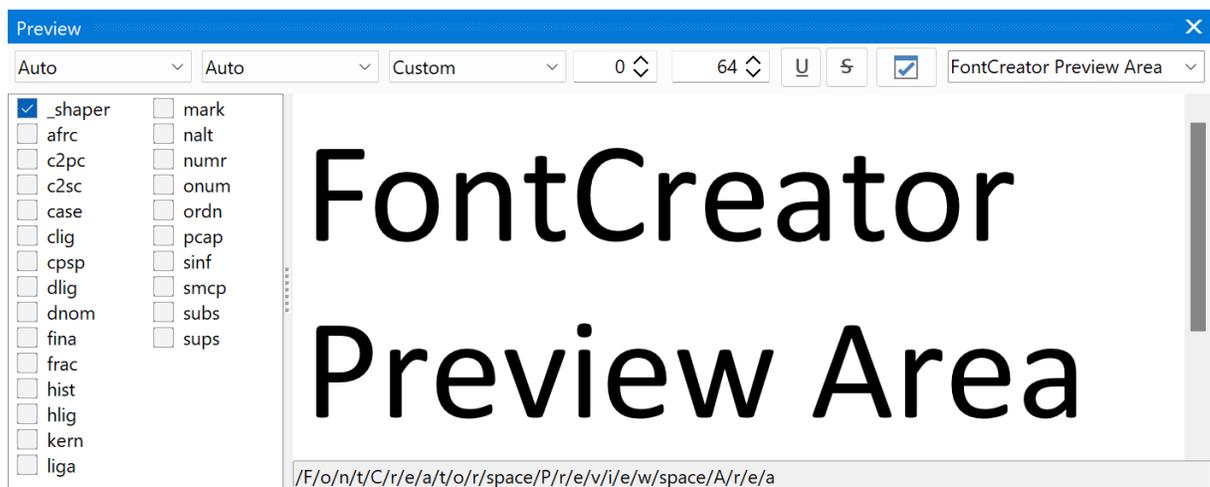
[Font Validation](#)

7.8 Preview

While editing a font, you can preview the results with the Preview panel. You can choose a standard text sample from the drop-down list or enter your own text. It can be toggled on and off using the **F8** shortcut key and is available in both the **Font** panel and the **Glyph** panel.

You can also use special formatting as explained in [Preview in FontCreator](#).

Selected glyphs in the font panel can be displayed in the **Preview** panel by pressing the **P** shortcut key. You can also use **Shift P** to add the selected glyphs to the current text.



The drop down lists available at the upper left and the first edit control after these lists are all related to OpenType layout features as defined in the OpenType Designer dialog. To test language specific features, select the corresponding script in the first drop down list, and then select the language in the second drop down list.

The edit control can be used to select which alternates to show.

To test your kerning, select the **kern feature** in the list of features. You can select multiple features so that you can see how they work together.

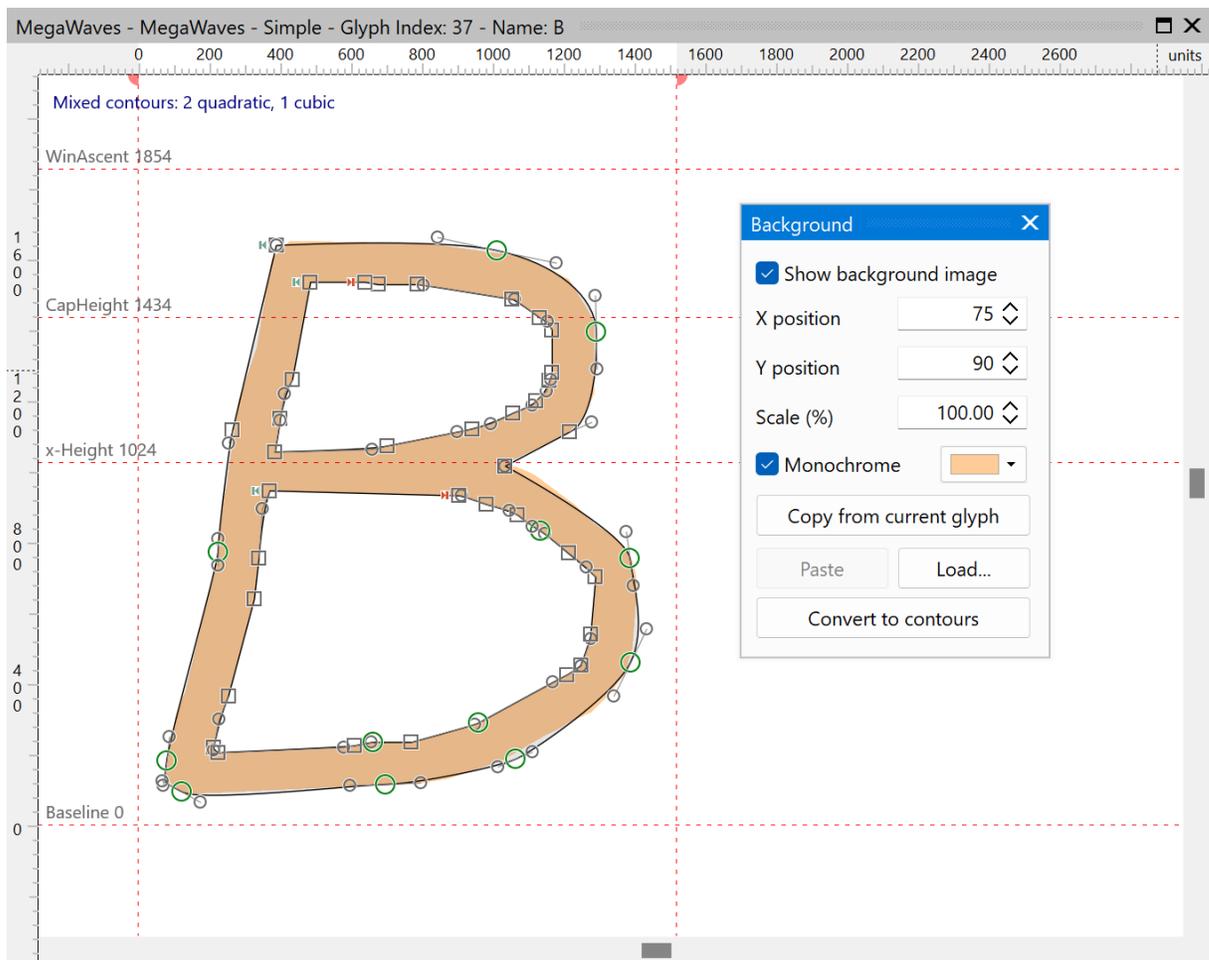
The **Preview** panel can be resized to show more text.

If you wish to edit a glyph's outline or adjust its metrics, click the specific glyph in the preview area to jump to it in the Font or Glyph panel.

Note: If the Glyph Display Mode (available from the View menu) is set to Auto or Color, then the foreground and background color come from the Palette panel.

7.9 Background Image

You can add a background image on a **Glyph** panel through the **Background** panel. It can help you with your glyph design, for example by reconstructing a scanned letter. It can be toggled on and off using the **F9** shortcut key, but is available only in the **Glyph** panel.



Click the **Copy from current glyph** button to place a copy of the current outline as the background image. To add a background image, paste an image from the clipboard, or click the **Load** button and select the image that you want to use. Use the position and scale fields to move and scale the image. Check the **Monochrome** box, to show the background image in one color.

The **convert to contours** button converts the raster based image into vector-based contour which will become part of the existing glyph outline.

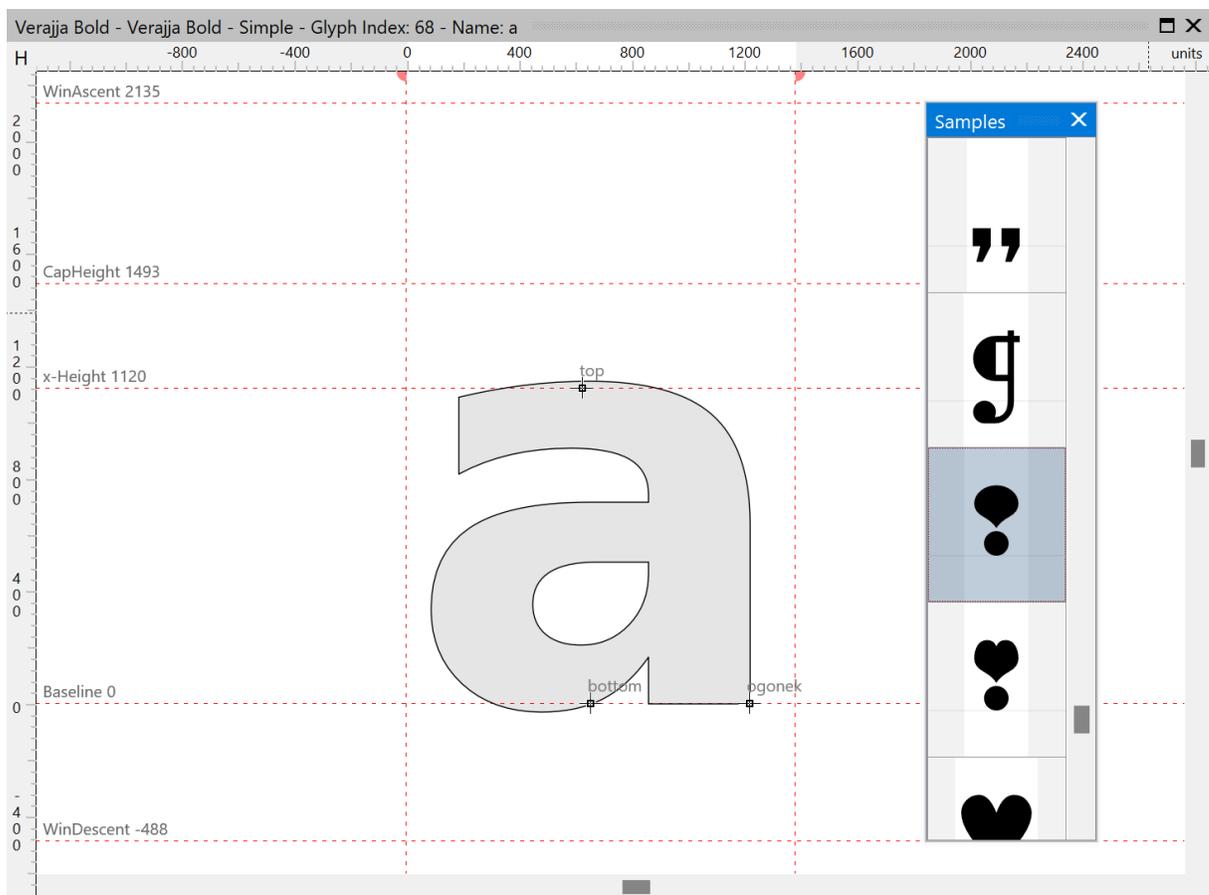
7.10 Samples

The **Samples** panel provides a powerful way of adding some contours you may frequently want to use in fonts or in glyphs.

You can use any of these glyphs in your own fonts. The outlines are royalty free. Credits go to [Bhikkhu Pesala](#).

With the **Samples** panel you can drag and drop a sample glyph to a cell on the **Font** or **Glyph** panel. The panel can be docked or floating, and can be toggled on and off using the **F12** shortcut key.

Note: When you drop a sample onto a composite glyph on the Font panel it will be replaced with a Simple Glyph. To drop a sample into the glyph panel of a composite glyph, convert it to a simple glyph first. The sample will be added to the existing contours.



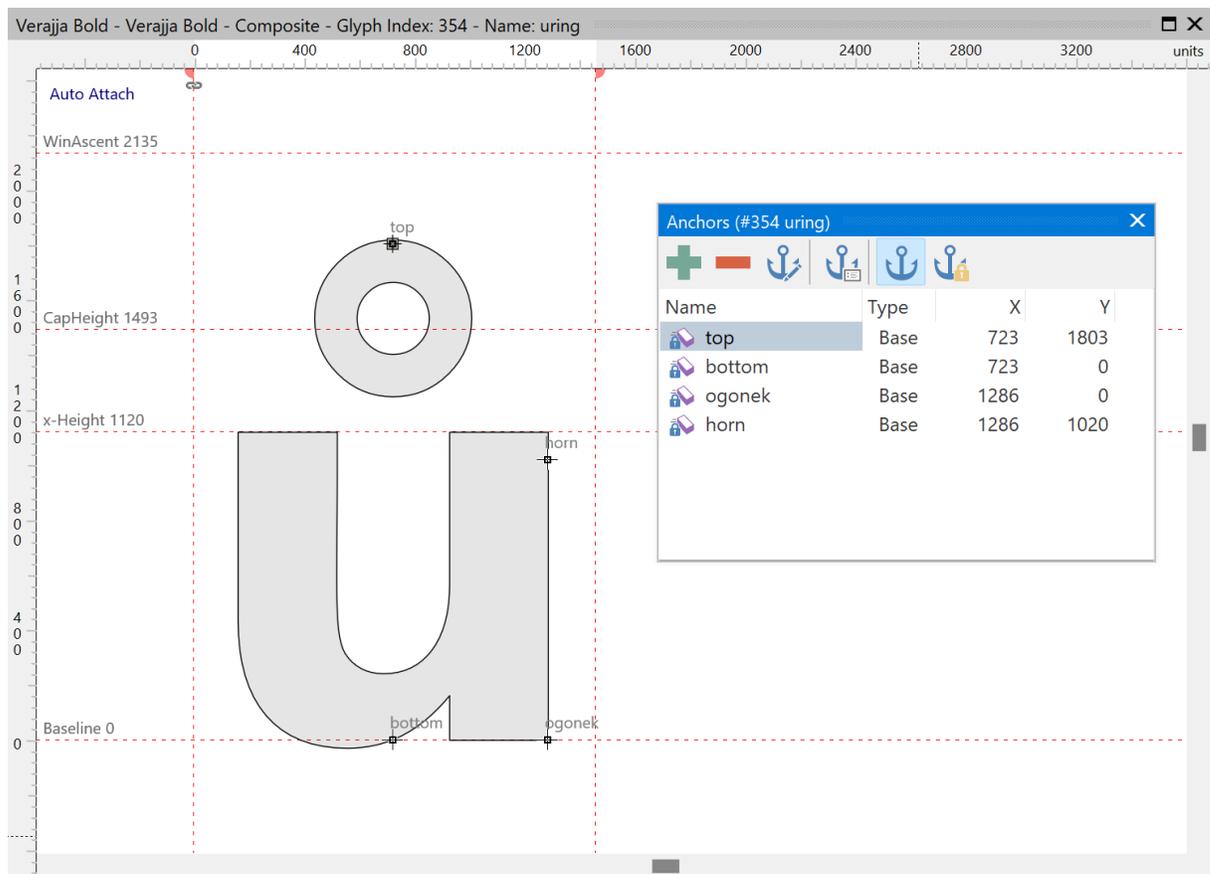
You can change the samples by making a special samples font, which includes your own samples. From the **Tools** menu you go to **Options** and go to the **General** tab. Here you can select the font filename to be used in the **Samples** panel.

7.11 Anchors

The Anchors panel gives you a quick overview of the anchors defined for a glyph. Anchors are used by Auto Attach glyphs and within these OpenType layout features:

- Mark-to-Base lookups
- Mark-to-Mark lookups
- Mark-to-Ligature lookups
- Cursive Attachments lookups
- To define caret positions for ligatures

With the first three lookups the anchors define where glyphs are stacked onto each other, and with cursive attachments the special Entry and Exit anchors are used to place two glyphs next to each other. Add a caret anchor to a ligature to define where a client should select and highlight ligature components in displayed text. We are not aware of any client who actually uses the ligature caret data, but this might change in the near future. You should add a caret anchor between the individual base components, so use one for the "fi" ligature, and two for the "ffi" ligature.



When an anchor is in use by the OpenType layout features an icon with an arrow will appear in front of it. If the anchor is inherited from another glyph because it is using [Auto Attach](#), the icon will show a lock as you can't move or delete such anchors.

Move Anchor

To change the anchor position, you can drag it in the glyph panel. To move only horizontally or vertically, press Shift as you drag the anchor. Press Alt to ignore the snap to grid and snap to guidelines features.

Add Anchor

Add a new Anchor. To add an anchor at a specific position, right-click inside the Glyph panel, and select Add Anchor.

Delete Anchor

Deletes the selected anchor

Edit Anchor

Edit selected anchor position. You can also double-click on the entry in the list.

Anchor Manager

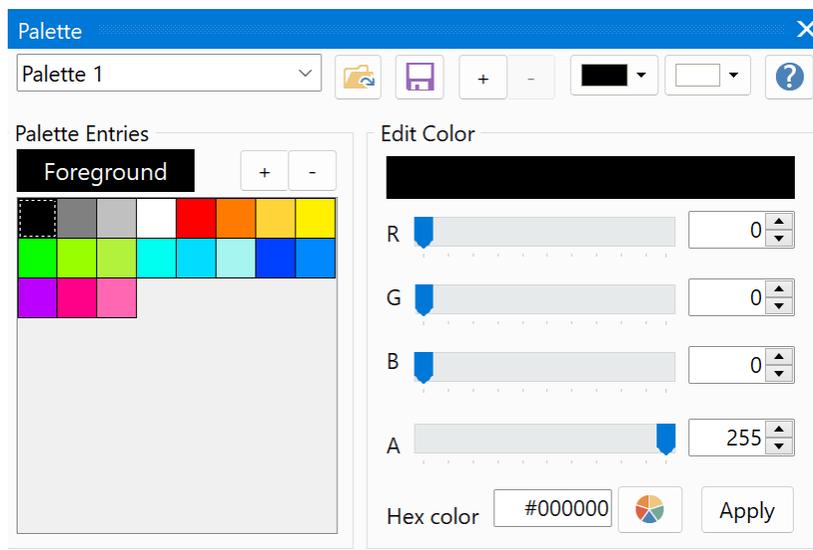
Add/Delete/Modify Anchor classes

Tip: use [Complete Composites](#) Anchor Based to automatically create composite glyphs based on anchors.

Note: [Auto Attach](#) also makes use of Anchors.

7.12 Palette

The Palette panel allows you to edit the palettes and the colors used in your font.



The palette drop down list shows the active palette. You can add and/or remove palettes by clicking the + and - buttons. Note that there must be at least one palette.

The foreground and background color selectors allow you to set the foreground and background colors that will be used in FontCreator's Preview panel. The background color is also used with color glyphs within the Glyph and Font panels. Note that each palette has its own foreground and background colors.

The Palette entries list the current available colors for the active palette. You can modify a color by selecting the color and using the sliders, edit fields or the color

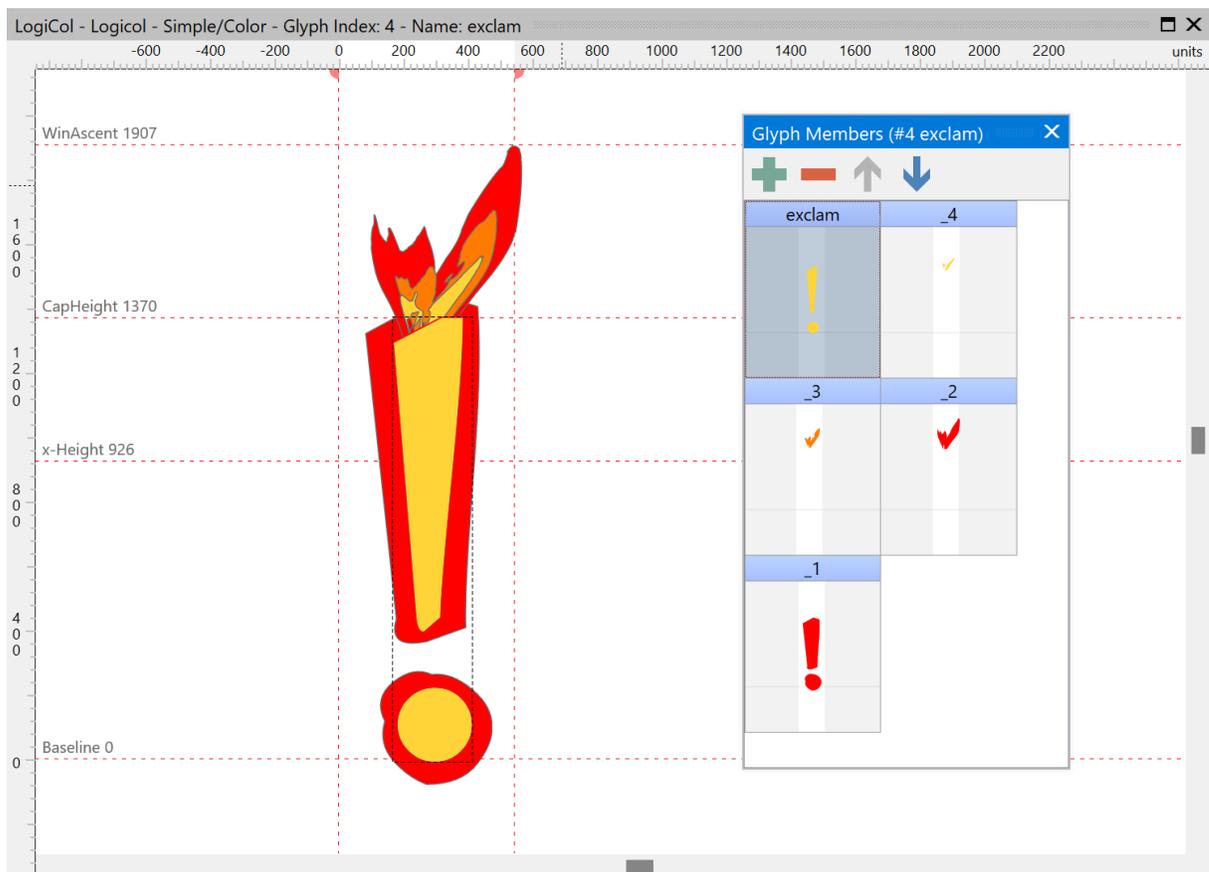
chooser. Use the **Apply** button to process the changes. You can use the + and - buttons to add or remove colors. Note that only unused colors can be removed.

The **Foreground** palette color is a special color that depends on the font color that the user or host application has set as the active font color.

The load and save options allow you to easily exchange the complete set of palettes between fonts.

7.13 Color Glyph Members

The Color Glyph Members panel gives you a quick overview of the glyphs used in a color glyph and allows you to add and remove glyphs. When glyph members overlap then the order in which they appear becomes important. You can change the glyph member order by using the **Move Up** and **Move Down** buttons.



Part

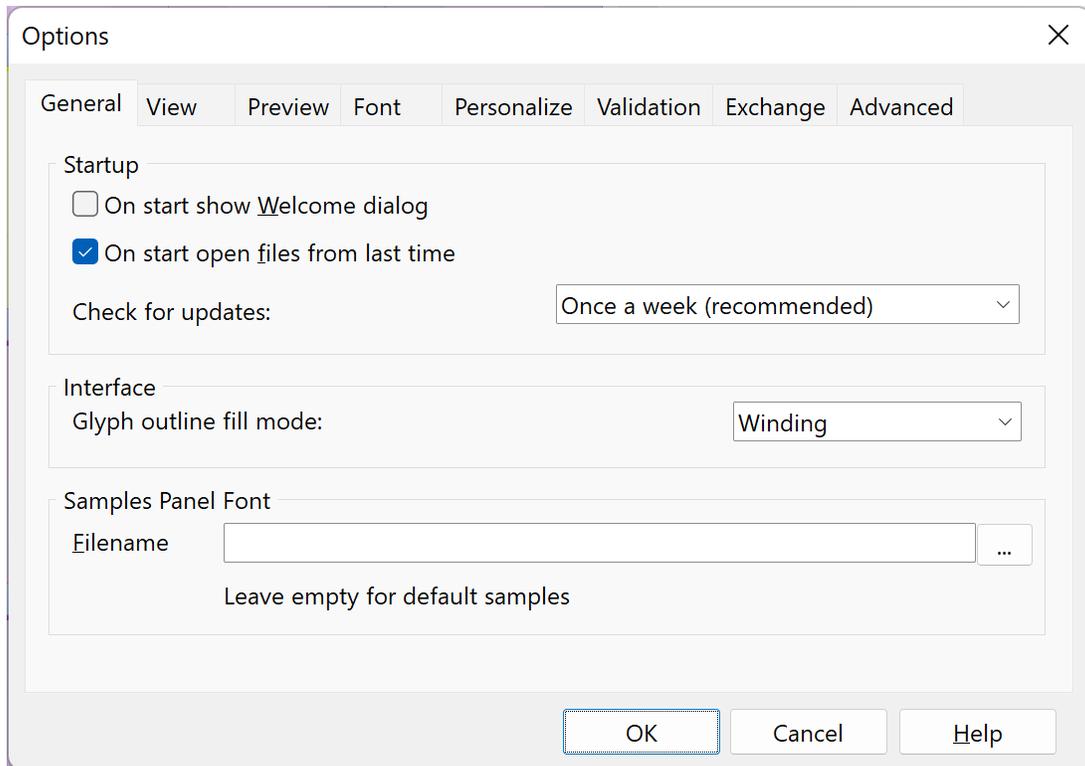


8 Customizing FontCreator

8.1 Options

8.1.1 General

You can customize the way fonts are loaded, saved and shown with the **Options** dialog (Select **Options** from the **Tools** menu).



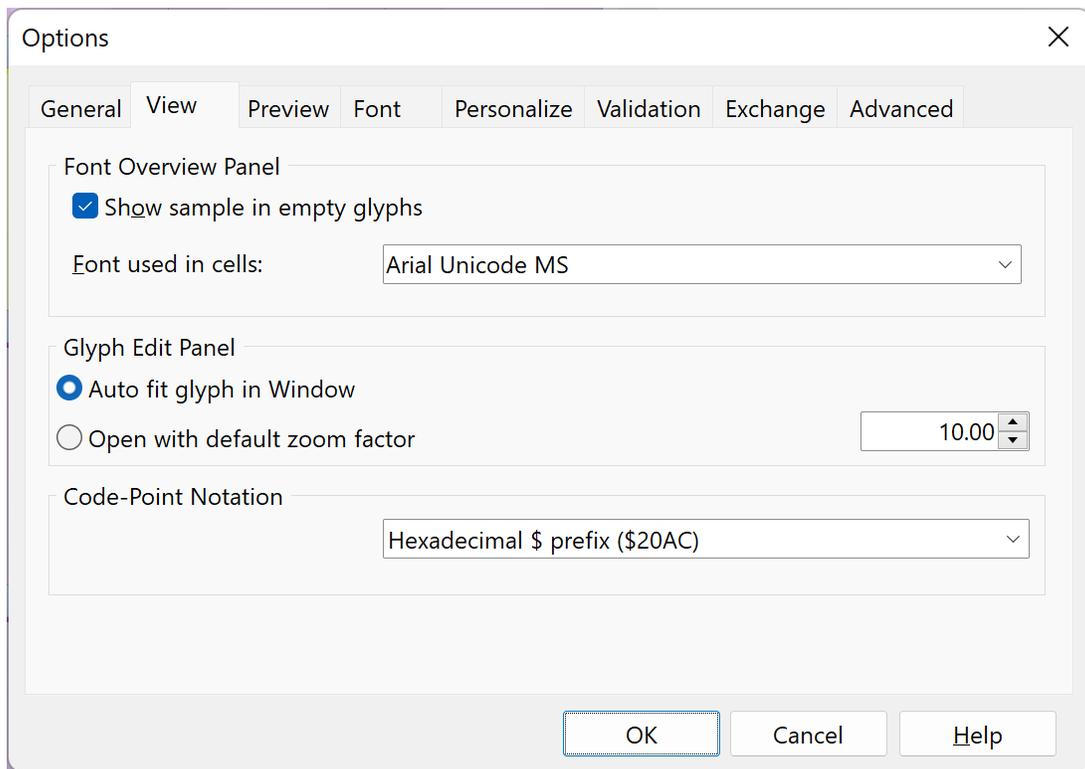
Use the fields on the **General** page when you want to change the welcome dialog, the interface settings or the **Samples panel** font.

Check for updates

Set the interval that FontCreator uses to check for updates. An interval of Once a week will ensure you are always working with the latest version of FontCreator.

8.1.2 View

On the **Tools** menu, click **Options**, and then click the **View** tab. Here you can adjust the **Font** and **Glyph** panel settings.



Show sample in empty glyphs

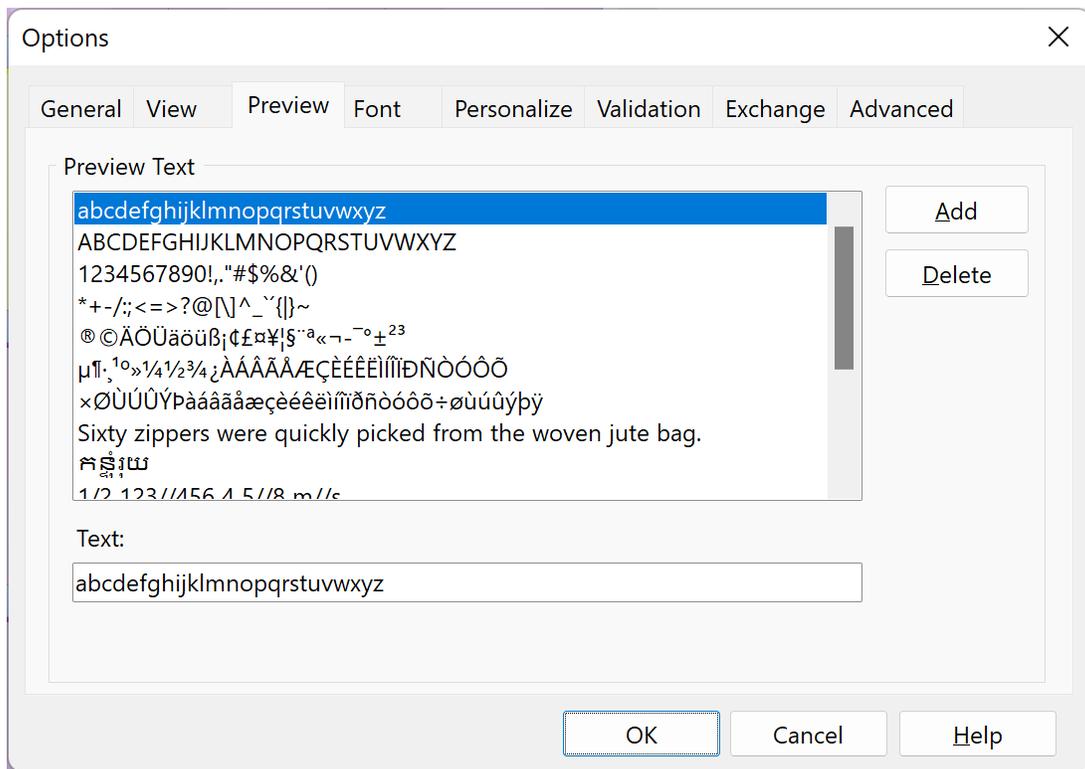
Will show a grey sample glyph for empty glyphs in the font panel. You can change the font to use by selecting one from the installed fonts list.

Glyph Edit Window

Here you can set the way a **Glyph** panel will show the glyph

8.1.3 Preview

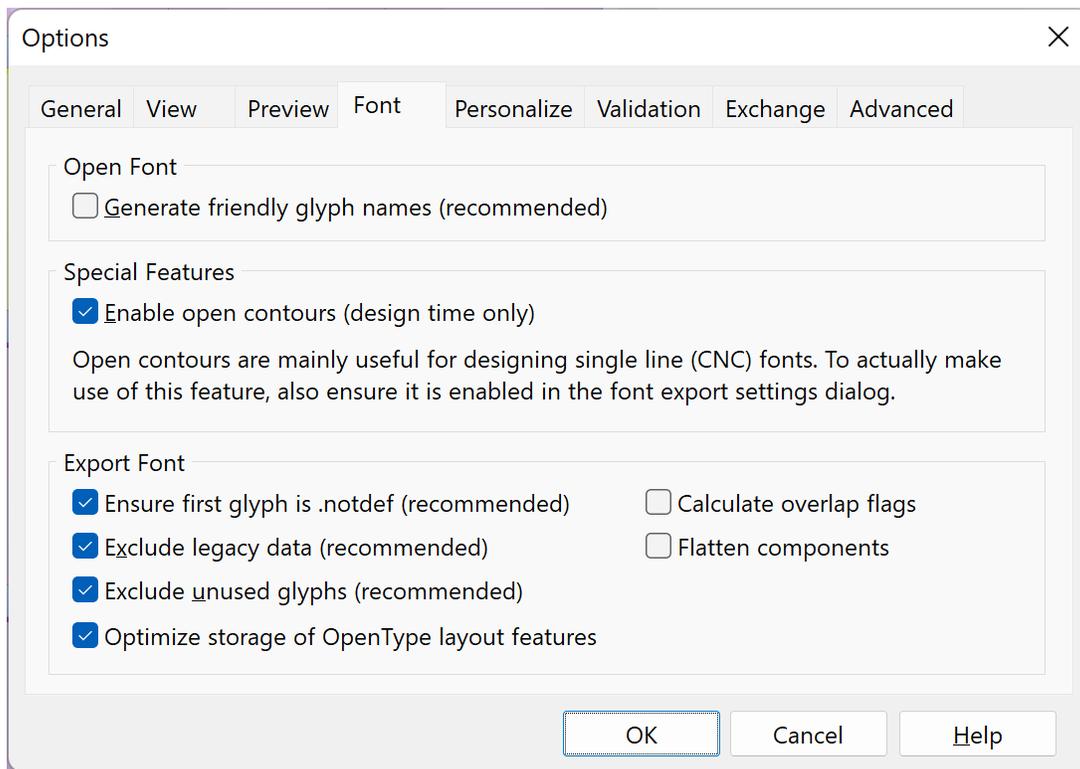
On the **Tools** menu, click **Options**, and then click the Preview tab.



Here you can maintain your preview text, which is used inside the Preview panel.

8.1.4 Font

On the **Tools** menu, click **Options**, and then click the **Font** tab.



Generate friendly glyph names

This option will generate friendly glyph names on opening a font file. Some fonts contain invalid or awkward glyph names, while more recent fonts no longer come with glyph names. FontCreator will try to generate glyph names that best represent the glyphs. It is recommended to set this option, but if you use more tools to edit your fonts, it might be best to keep glyph names untouched.

Enable open contours

This option will allow you to design single line fonts. Such fonts are mainly used for engraving, so such fonts usually look strange in regular software. If this option is enabled, the [Font Export Settings dialog](#) will show an additional option, Open Contours.

See [Single Line Fonts](#) for more information.

Ensure first glyph is .notdef

All fonts must include a .notdef (missing character) glyph as first glyph. This option ensures the first glyph is the .notdef glyph with a standard rectangular outline.

Exclude legacy data

By default FontCreator will only store naming fields in the Windows platform, which is used cross-platform nowadays. If your font must also include legacy naming fields for the Macintosh platform, then uncheck this option. Optionally exclude legacy cmap data. We recommend to test your font to be sure it works on all older systems you want to support.

Exclude unused glyphs

Unused glyphs are not accessible in a normal manner. This option ensures such glyphs are excluded.

Optimize storage of OpenType layout features

This will store the OpenType layout feature data in an optimal compressed way.

Calculate overlap flags

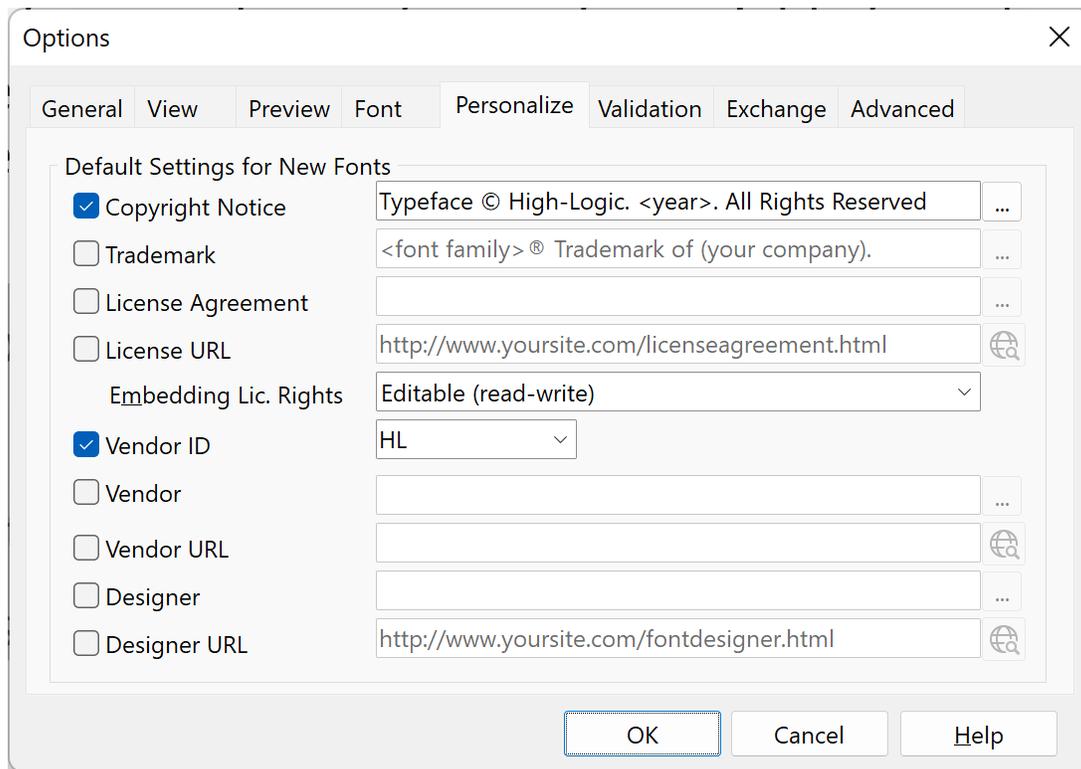
On exporting a font, FontCreator can include a flag for each glyph that has overlapping contours or overlapping composite glyph members. This flag is not required and it takes some time to calculate these overlaps, so only enable this if there is a specific need for it.

Flatten components

Composite that make use of other composites will be flattened, so composite glyphs will only make use of non-composite glyphs. Olders versions of Microsoft Word have issues with nested components. This option solves that issue.

8.1.5 Personalize

On the **Tools** menu, click **Options**, and then click the **Personalize** tab.

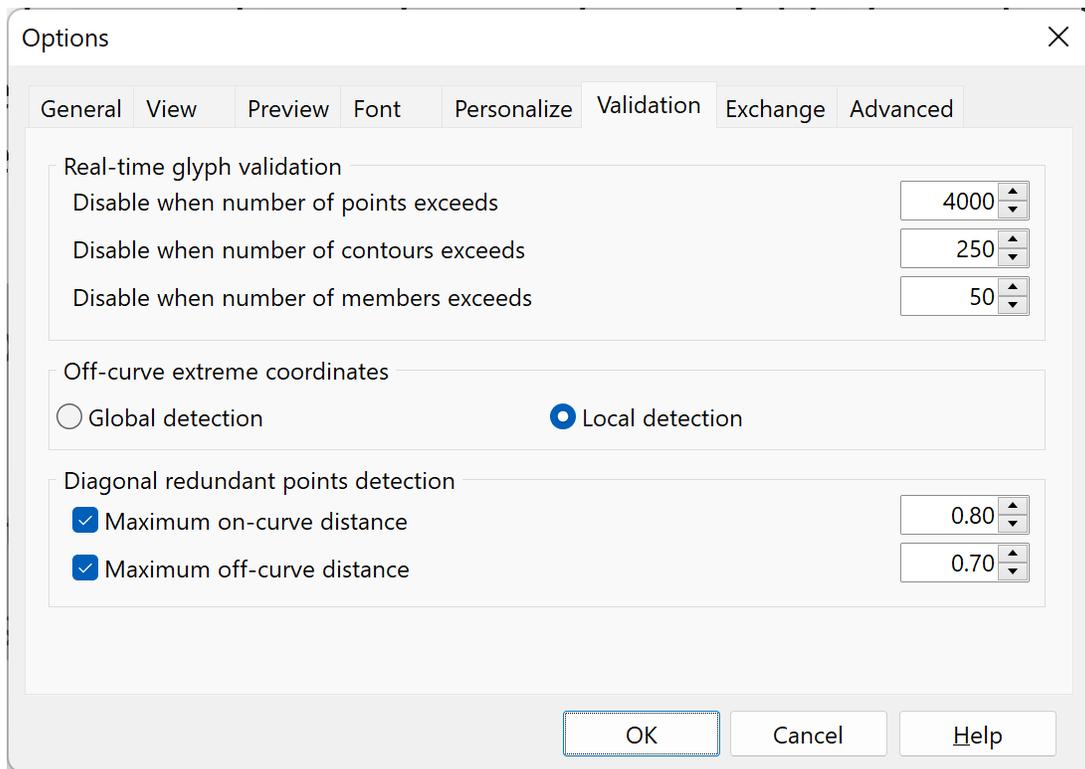


Default Settings for New Fonts

These fields are used as initial naming fields when a new font is created. <year> will be replaced by the current year and will be replaced by the actual font family name.

8.1.6 Validation

On the **Tools** menu, click **Options**, and then click the **Validation** tab.



The Real-time glyph validation settings are used to determine whether the real-time glyph problem validation should be disabled as it would be too time-consuming. These settings are only used by the real-time glyph problem validation. When disabled, use the Refresh button on the [Validation panel](#) to revalidate the glyph.

Off-curve extreme coordinates detection can be performed globally or locally.

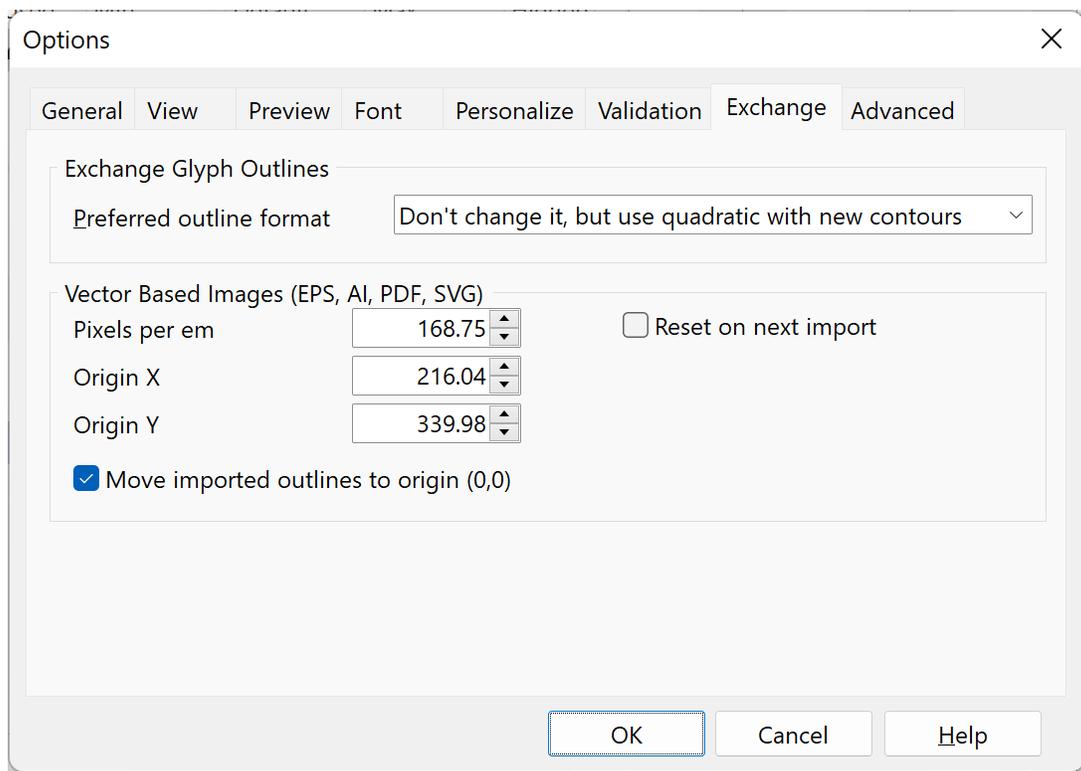
Note: The Validation features are not available in the Home Edition of FontCreator.

See also:

[Font Validation](#)

8.1.7 Exchange

On the **Tools** menu, click **Options**, and then click the **Exchange** tab.



Exchange Glyph Outlines

If you rely on other software to design your vector based outlines, then it is probably best to set Preferred outline format to "Don't change..." as that allows you to exchange contours without outline format conversion.

Vector Based Images

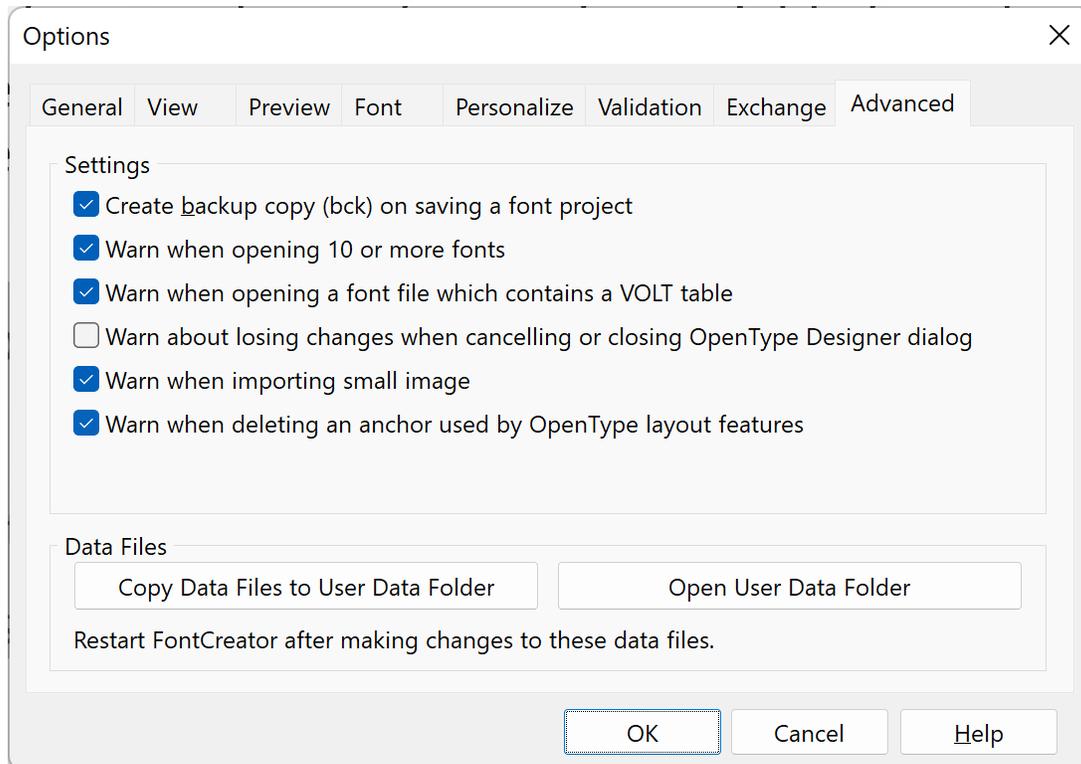
These settings are used when vector-based images are imported as well as when contours are copied to the clipboard, so they can be exchanged with vector based image editing software while maintaining scale and position. The preferred values depend on the used document, and can be determined from within your external vector software. Do ensure that you measure the values while units are shown in pixels (px). The height of your image is usually a good value for Pixels per em.

Check the **Move imported outlines to origin** option if you prefer to always place imported vector based images at position (0,0).

Check **Reset on next import** if you want to scale and position the next imported vector based image at the visible glyph outline area.

8.1.8 Advanced

On the **Tools** menu, click **Options**, and then click the **Advanced** tab.



Create backup copy (bck) on saving a font project

Copies the previous version of a font project as a backup copy every time you save a font project. Each new backup copy replaces the previous backup copy. FontCreator saves the backup copy (with a file name extension .bck) in the same folder as the original.

Warn when opening 10 or more fonts

This indicates and determines if a warning will be issued when opening 10 or more fonts. Opening a lot of fonts at the same time will require a lot of system resources.

Warn when opening a font file which contains a VOLT table

This indicates and determines if a warning will be issued when opening a font file which contains a VOLT table. You can then choose to either open, parse and use the VOLT data or use the available OpenType layout features.

Data Files

FontCreator uses several data files for advanced settings and customizations. Normally these files are stored in a system folder where they cannot be changed. If you want to edit these data files they must first be copied to your user data folder. First click **Copy Data Files to User Data Folder** and then **Open User Data Folder** to start editing the files. See [FontCreator data files](#) for more information.

8.2 Keyboard Shortcuts

The following keyboard shortcuts can be used to quickly accomplish frequent tasks:

General Shortcuts

Press	To
Ctrl + Tab	Next Window
Shift + Ctrl + Tab	Previous Window
Alt + Enter	Show Font Properties Panel
F1	Help
Ctrl + F2	Show/Hide Font Properties Panel
Shift + F2	Show/Hide Palette Panel
F3	Find Next
Shift + F3	Find Previous
F4	Show/Hide Glyph Properties Panel
Shift + F4	Show/Hide Anchors Panel
Ctrl + F4	Close Window
F5	Test Font as TTF/OTF
Ctrl + F5	Test Font as WOFF
F6	Show/Hide Transform Panel
Ctrl + F6	Show/Hide Masters and Layers Panel

Press	To
F7	Show/Hide Glyph Validation Panel *
Shift + F7	Show/Hide Status Bar
Ctrl + F7	Show/Hide Variations Panel
F8	Show/Hide Preview Panel
Shift + F8	Show/Hide Color Glyph Member Panel
Ctrl + F8	Show OpenType Designer/ Edit Lookup in Code Editor
F9	Show/Hide Background Image Panel
F10	Activates the Main Menu
Shift + F10	Shows the context menu
Ctrl + F10	Launch MainType
Ctrl + F11	Launch Windows Font Folder
F12	Show/Hide Samples Panel
Ctrl + F12	Launch Windows Character Map
Delete	Delete selection
Ctrl + A	Select All
Ctrl + C	Copy
Ctrl + E	Displays the Font Export Settings dialog
Ctrl + F	Displays the Find dialog
Ctrl + H	Toggle Fill Outlines
Ctrl + I	Open Installed Fonts
Ctrl + J	Set as Mask
Shift + Ctrl + J	Add to Mask
Ctrl + M	Swap Mask

Press	To
Shift + Ctrl + M	Clear Mask
Ctrl + N	New Project
Ctrl + O	Opens Font or Font Project
Ctrl + P	Print Glyph, Font, or Font Properties
Ctrl + S	Save Project
Ctrl + T	Invert Selection
Ctrl + U	Clear Selection
Ctrl + V	Paste
Ctrl + X	Cut
Ctrl + W	Close current window
Ctrl + Y (or Shift + Ctrl + Z)	Redo
Ctrl + Z	Undo
Shift + Ctrl + A	Export font in all formats
Shift + Ctrl + E	Export font as Desktop font (ttf/otf)
Shift + Ctrl + S	Save all projects
Shift + Ctrl + W	Export font as Web font (woff/woff2)
Shift + Ctrl + Z (or Ctrl + Y)	Redo

Keys for working in the Font panel

Press	To
Enter	Open Glyph panel
Backspace	Clear selected glyphs

Press	To
P	The first 255 characters mapped to the selected glyphs will be placed into the Preview panel
Shift + P	Append up to 255 selected glyphs to Preview panel
Ctrl + Plus Sign	Increase the cell size
Ctrl + Minus Sign	Decrease the cell size
Ctrl + Shift + Plus Sign	Increase character size
Ctrl + Shift + Minus Sign	Decrease character size
Ctrl + Mouse wheel	Change cell size
Ctrl + Shift + Mouse wheel	Change character size
Ctrl + 0	Reset character and cell size to their defaults
Ctrl + 1-5	Toggle glyph tag 1-5
Ctrl + L	Toggle category panel
Ctrl + Alt + V	Paste Special

Keys for working in the Glyph panel

Press	To
Left Arrow	Pan Left (nothing selected) Move selection 10 funits left
Right Arrow	Pan Right (nothing selected) Right Move selection 10 funits right
Plus Sign or Ctrl + Plus Sign	Increase the zoom factor
Minus Sign or Ctrl + Minus Sign	Decrease the zoom factor

Press	To
Ctrl + 0	Reset zoom factor to default
Ctrl + 1-5	Toggle glyph tag
Alt + Left Arrow	Go to previous glyph
Alt + Right Arrow	Go to next glyph
Delete	Delete selection (e.g. point, contour, or glyph member)
Backspace	Delete selected on-curve point and adjust curve
;	Set default bearings
N	Change selected points to on-curve
F	Change selected points to off-curve
G	Add guideline through two selected nodes, or add orthogonal guidelines at a single selected node
A	Add points after selected points
H	While pressed down, hides everything except the glyph
Q	Select previous contour/point index or composite glyph member. Use Shift to add to current selection.
W	Select next contour/point index or composite glyph member. Use Shift to add to current selection.
P	Toggle between point and contour mode
S	In contour mode, toggle between scale and skew and rotate. In point mode toggle between move and scale.
T	Text
C	Toggle between standard and color mode
K	Knife (not in color mode)

Press	To
M	Measure
B	Paint bucket (only in color mode)
Z	Toggle between Zoom to default and Zoom to selection
/	Fit to Window
Middle mouse button (or space+Left mouse button)	Pan around the Glyph panel
Double-click	While editing an empty or simple glyph, this will switch contour/point mode While editing a composite glyph, this will open the composite glyph member properties window
Double-click on rulers	Edit guidelines
Double-click on guideline	Adjust guideline
Shift-click on guideline	Rotate guideline
Ctrl + Drag guideline	Duplicate guideline
Click on selected contour	show/hide rotate and skew handles
Shift + Drag	Constrain movement, scaling, skew, and rotation
Ctrl + Drag	Duplicate selected contour(s) or composite glyph member(s)
Alt + Drag	Disable Snap

Keys for working in the OpenType Designer window

Press	To
Ctrl + L	Toggle OpenType script panel

Press	To
Ctrl + F8	Edit selected lookup in Code Editor
Insert	Add new substitution to selected lookup table
Delete	Delete selected substitution from lookup table

You can reset all toolbar positions and sizes by pressing the Ctrl key while starting FontCreator.

* Not available in the Home Edition of FontCreator.

8.3 FontCreator Data Files

FontCreator uses several extra data files to read settings and other program specific information. These should not be edited directly, but copied to the windows user data folder. This can be easily done through the Options dialog on the [Advanced](#) tab.

Unicode/Blocks.txt	Standard Unicode database file containing the Unicode blocks
Unicode/UnicodeData.txt	Standard Unicode database file containing all the Unicode glyph information
ArabicShaping.txt	Standard Unicode database file containing information about Arabic shaping
glyphlist.dat	The Adobe glyph list
tags.txt	Controls the names of the five tags that can be assigned to each glyph
preview.txt	Contains the standard preview texts for the Preview panel
glyphnamesnew.dat	Contains overrides of the default friendly glyph names, as used when opening existing fonts and when you generate glyph names on the Glyph Properties dialog
Transform folder	Contains all transform scripts that can be used with the Glyph Transformer

WOFFTest Contains the files used for testing your fonts in your web browser

Note: FontCreator only reads these files at startup. This means that after you have changed any of the files, you must restart FontCreator for the changes to take effect.

Part



9 About Fonts

9.1 TrueType

TrueType is a scalable font technology designed by Apple Computer in the late 1980s. Apple and Microsoft then worked together to push the technology into the market as alternative to Type 1 from Adobe. Apple implemented the font system and Microsoft the printing engine.

A TrueType font file contains data, in table format, that comprises an outline font. Rasterizers use combinations of data from the tables contained in the font to render the glyph outlines. The TrueType format only supports glyph outlines with quadratic bezier curves and lacks support for layout features.

9.2 OpenType

In 1997 the OpenType font specification was released as an extension to the TrueType format. OpenType was developed jointly by Microsoft and Adobe to end their font war. It comes with support for PostScript Type 1 data and advanced layout features, which allow font designers to design better international and high-end typographic fonts.

Even though OpenType fonts can contain either PostScript Type 1 or TrueType outlines, it is a common mistake to talk about TTF and OTF to diverse between the two outline formats.

TrueType has been superseded by the OpenType format, but people still refer to TrueType fonts, while in fact 99.9% of all fonts that come with Windows are OpenType fonts.

See also: <http://forum.high-logic.com/viewtopic.php?f=5&t=1619>

FontCreator fully supports OpenType fonts. You can either draw your glyph outlines with quadratic or cubic curves. Use the [OpenType Designer](#) to add advanced OpenType layout features to your fonts.

More information about the OpenType font specification:

<https://docs.microsoft.com/en-us/typography/opentype/spec/> 

9.3 Variable

In September 2016 variable font technology has been added to OpenType specification version 1.8.

A variable font is a single file, containing one or more axes (for example weight or width), which allow you to use a continuous range of style variations by means of interpolation.

FontCreator has all the tools to [make variable fonts](#).

9.4 Color Extensions

There are two scalable color extensions.

- COLR; initiated by Microsoft, using internal tables COLR and CPAL
- SVG; initially proposed by Mozilla and Adobe, using a SVG table (optionally along with CPAL)

COLR (and required CPAL)

As revealed at the Microsoft Build Developer Conference in June 2013, Windows 8.1 comes with a revolutionary extension to the OpenType font standard, which introduces multi-color fonts. The technology which is both simple and powerful uses multi-layer glyphs which are in essence scalable outlines that are rendered and processed like any other character, except each layer has its own color using internal tables COLR and CPAL.

High-Logic is convinced this new technology will open new colorful doors. That is why FontCreator was the first font editor to support the new multi-color fonts extension.

The beauty of the color extension is that the fonts will continue to work like any other font on devices and systems that don't support the extension yet. That is why it is strongly recommended to always include the base outlines for each glyph (used as fallback in case color fonts are not supported, or not wanted in a particular situation). This is currently the only color extension that works with variable fonts.

In December 2021 OpenType Specification version 1.9 was released. It includes a major improvement to COLR, known as COLRv1. It allows you to make color fonts with gradient fills, as well as more complex fills using other graphic operations, including affine transformations and various blending modes. FontCreator currently does not support COLRv1.

SVG (and optional CPAL)

This extension uses scalable vector graphics (SVG) documents. There is no visual SVG editor in FontCreator, but you can generate the SVG color format out of the COLR extension. You can also import an SVG document for each glyph.

Which one to choose?

With the rise of COLRv1 the color font war is most likely won by Microsoft, so we recommend to use COLR. You can then also automatically export to SVG without any additional design efforts. COLR is supported by all major web browsers, and more and more Windows software like Word, and paint.net.

Most Adobe software support OpenType SVG color fonts, but the SVG extension is not supported in Chromium-based browsers, like Edge, Chrome, Opera, and Vivaldi, and there are no plans to support it.

Support for Color font extensions

Not all software and web browsers support the scalable color extensions. We have gathered a more complete list in our online color tutorial:

<https://www.high-logic.com/font-editor/fontcreator/tutorials>

9.5 Web Open Font Format (WOFF)

The Web Open Font Format (WOFF & WOFF2) is a font format based on the same principles as OpenType and TrueType fonts (also known as desktop fonts) but has been optimized for use in web pages. The format is supported across all recent major browsers. Fonts that are used in web pages are also called web fonts. Those fonts come from the web and are temporary loaded within the web browser.

FontCreator supports both importing and exporting WOFF and WOFF2 fonts.

More information about Web fonts and how to use them:

<http://www.webfonts.com> 

More information about WOFF File Format 1:

<http://www.w3.org/TR/WOFF/> 

More information about WOFF File Format 2:

<https://www.w3.org/TR/WOFF2/> 

9.6 FontCreator Project

FontCreator uses its own proprietary format to store the font data, several other settings and configuration parameters:

- Font data
- Font properties
- Glyph outlines, including optional color
- Export settings
- Guideline, Grid and Metrics settings
- OpenType Layout Features
- User notes

9.7 Unified Font Object (UFO)

The Unified Font Object (UFO) is a cross-platform, cross-application, human-readable format for storing font data. It is stored within a folder, that consist of metainfo.plist, fontinfo.plist, lib.plist, and a collection of individual glyph outlines along with contents.plist. The folder may contain other files like features.fea, but those are not required.

FontCreator supports both importing and exporting UFO based fonts. UFO import covers UFO versions 1, 2, and 3. Exporting UFO is always version 3. FontCreator does support kerning, feature.fea, and several public libs.

Supporting libs:

- public.glyphOrder
- public.skipExportGlyphs
- public.unicodeVariationSequences
- public.verticalOrigin
- public.markColor
- public.openTypeMeta
- com.github.fonttools.varLib.featureVarsFeatureTag

FontCreator currently does not support other extensions or binaries, so if external software has custom data, that will be ignored on importing.

Note: UFO does not cover all font data. We recommend to always save your fonts as a [font project](#).

More information about UFO:

<http://unifiedfontobject.org/> 

See also:

[OpenType Layout Feature Code Editor](#)

9.8 Designspace

A designspace can combine several UFOs to make a variable font. It describes axes, masters (known as sources), instances, basic feature variation (rules), STAT information, and optionally the (subsetting) variable fonts that can be generated.

FontCreator supports both importing and exporting designspaces. It supports designspace format version 5.0 as well as previous versions. FontCreator fully supports these new subset capabilities:

- Define multiple subsets
- Change the default location
- Mark an axis as discrete
- Extrapolate axes

We recommend to use a designspace in case you want to exchange a variable font between font editors.

9.9 Glyphs File Format

The glyphs file format allows you to exchange your fonts with people who use a different font editor. However like any other file format, it does not cover everything.

FontCreator can open all known versions of the format, but some information will be lost during the import. FontCreator always exports as glyphs version 3, but it does not fully support all data.

Our own FontCreator Project font file format is able to store all font data that FontCreator supports, so in general it is best to use that as your main source. From there, you can export fonts for use on desktop or web, as well as to interchangeable

font project resources. This will make sure that regardless of the font format you export, all information about the font will remain available.

9.10 Font Copyright

Unless you know otherwise, you should assume all fonts to be copyrighted works that are someone's property and treat them as you would any other software. Fonts are software products in their own right, and are protected by international copyright law as well as individual license agreements. Even redistributing so-called freeware or public domain fonts is problematic. If you have created a font yourself (without using anything from other fonts), it is your property.

The use of any commercial font is governed by the terms of its manufacturer's End User License Agreement (EULA). Several major font vendors specifically allow altering a font, as long as the altered font is only used on machines for which you have licensed the original font. If you have questions about what can or can't be done with a font, you should contact that font's manufacturer.

The Copyright field in the **Legal** area on the **Font Properties** panel may direct you to the copyright holder, but be aware that this field may be blank, or may have been altered. Also the Trademark, Description, License Agreement, and the License URL fields might have important information.

Index

- . -

.notdef 46

- A -

Adobe OpenType Feature Description Language 135
Advance Width 57, 166, 246
Advanced 272
Anchors 124, 126, 127, 259
Ascent 217
AutoKern 168
Autokerning 118

- B -

Background Image 256
Backup 272
Baseline 156
Bearings 57
Bezier 63, 71
Bézier 63, 71
Bitmap to Outline 172
Bold 42, 165, 211

- C -

CapHeight 156, 217
Caption 27
CFF 20
Changes 11
Character coverage 202
Character Map 185, 197
Character Ranges 240
Characteristics 233
Characters 34
 Insert 34
CharMap 185, 197
Class Manager 115
Classes 115
Close Font 24

CNC Font 51
Code page 240
Code Page Character Ranges 202
Color 85, 283
Color Glyph 86
Colorize
 Glyphs 85
Colors 88
COLR 86, 283
Commands 200
Compact Font Format 20
Complete Composites
 Anchor Based 82
 Composite Data 82
Composite Glyph 78
Composite Glyph Member Properties 80
Composites 82
Contour 56, 63, 66
 Direction 63
 Mode 63
Convert 172, 174
 Composite Glyph 78
Copy Glyph 38
Copyright 268, 288
Copyright Notice 202
CPAL 283
Create Font 18
Cubic 20
Curve 63
Customize 264

- D -

Descent 217
Design languages 202, 240
Designer 202, 268
Designspace 287
Direction 251
Duplicate 251

- E -

Ellipsis 46
Embedding 44
Emoji Variation Sequences 180
Empty Glyph 63
Empty Glyphs 264

Engraving Font 51
 Exchange 270
 Exclusion 75
 Export
 Desktop Font 20
 Settings 20
 Web Font 20
 Expression 57
 Extend Character Set 165
 External Tools 185
 Extreme 251

- F -

Fallback Font 50
 Family 211
 FEA 135
 Features 10
 New 11
 File Name 16, 41
 Find 33
 Font
 Information 161
 Single Line 51
 Single Stroke 51
 Table 163
 Font embedding 202
 Font Family 42
 Font matching 233
 Font name 41, 202
 Font overview 27, 264
 Font Style 211
 Font Type 42
 FontCreator
 Buying 13
 Editions 13
 Getting Help 13
 Getting Started 12
 Manual 13
 Overview 12
 Fonts
 Color 283
 Fonts folder 185
 Freedraw 77
 Full Font Name 211

- G -

Generate Composite
 Formula 36
 Generate Glyph Names 29
 Glyph 56
 Composite 56
 Empty 56
 Insert 36
 Mappings 246
 Properties 246
 Simple 56
 Type 27, 56
 Glyph Member 79
 Glyph Metrics 57
 Glyph Name 246
 Glyph names 266
 Clear 29
 Edit 29
 Search and Replace 29
 Glyph Transformer 165
 Grid Options 159
 Guideline 155
 Guidelines Options 153

- H -

Hinting 163
 Hybrid Glyph 85

- I -

Ideographic Variation Sequences 180
 Image to Outline 172, 174
 Import 270
 Import Image 172, 174
 Installing Fonts 196
 International 193
 Intersection 75, 251
 Italic 42, 165
 Italic Angle 211

- J -

Join Contours 75

- K -

Kerning
 Automatic 170
Kerning Pairs 121
Keyboard 193
Keystroke 197
Knife 75

- L -

Language 193
Last Resort Font 50
Layered 86
Layout 193
Left Side Bearing 156, 166, 217, 246
Left side-bearing 57
Legacy font data 266
Legal information 202
License 13
License Agreement 202, 268

- M -

Manual
 PDF 13
Mappings 249
Marks 124
Menu 200
Metrics 80, 217
Metrics Options 156
Microsoft Visual OpenType Layout Tool 135
Monospaced 45

- N -

Naming 268
Nonspacing combining marks 57

- O -

Off Curve 71
On Curve 71
Open Contour 66
Open Font 16

OpenType 282
OpenType Designer 95
 Autokerning 118
 Settings 117
OpenType Features
 Cursive Attachment 126
 Mark to Base 124
 Mark to Ligature 124
 Mark to Mark 124
OpenType Font Collection 183
OpenType Layout Feature Definition 135
OpenType Layout Features 92, 135
 Alternates 119
 Anchor Based 99
 Anchor Manager 127
 Basics 138, 144
 Chained Context 127
 Comments 139
 Cursive Attachement 99
 Examples 148
 Feature 141
 Figures 99
 Fractions 99
 Generate 99
 Group 141
 Initial Forms 99
 Language 140
 Ligatures 99, 119
 Lookup 143
 LookupFlags 144
 Multiple substitution 119
 Ordinals 99
 Ornaments 99
 Pair Adjustment 121
 Petite Capitals 99
 Pos 146
 Script 139
 Single Adjustment 120
 Single substitution 119
 Small Capitals 99
 Sub 145
 Substitutions 119
 Supported substitutions 92
Optical Metrics 166
Optimize Contours 66
Options 264
OTC 183
OTLFD 135

- P -

Palette 85, 88
 Colors 261
 Panose 233
 Paste Glyph 38
 Personalize 268
 Point 71
 Mode 63
 PostScript Name 211
 Preview 255, 262, 265
 Print
 Font 185
 Glyph 186
 Project
 Close 24
 File 16
 Open 16
 Save 19
 Proportional 45

- Q -

Quadratic 20
 Quotation Mark 46
 Quotes 46

- R -

Raster Image 172
 Redundant 251
 Register 13
 Revision 211
 Right Side Bearing 156, 217, 246
 Right side-bearing 57
 Rotate 66

- S -

Samples 257, 264
 Samples font 264
 Save Font 20
 Search 33
 Select 56
 Shortcut 273
 Simple Glyph 63

Single Line Font 51
 Single Stroke Font 51
 Smooth Curves 71
 Sort 182
 Split Contours 75
 Standardized variation sequences 180
 SubFamily 211
 SubTable Manager 116
 SubTables 116
 Support 13
 Supported languages 202, 240
 SVG 283
 Symbol 45

- T -

Table 163
 Ordering 163
 Tabular figures 57
 Tags 152
 Template 18
 Test
 OpenType Font 194
 TrueType Font 194
 WOFF 196
 WOFF2 196
 Test Desktop Font 194
 Test Web Fonts 196
 Text Samples
 Add 195
 Delete 195
 Edit 195
 Toolbars 200
 Trademark 202, 268
 Transform 165, 244, 246, 250
 TrueType 282
 TrueType Font Collection 183
 TTC 183
 Typeface 42
 Typo Ascender 217
 Typo Descender 217

- U -

UFO 286
 Unicode 45
 Unicode Character Ranges 202

Unicode Variation Sequences 180
Unified Font Object 286
Union 75
Unique Font Identifier 211
Units per em 202
Unused Glyphs 27
Updates 272
Used Glyph 33

- V -

Validation 186, 251, 269
 Category 189
 Results 189
Vector 270
Vector Image 174
Vector to Outline 174
Vendor 202
Version 202, 211
VOLT 92, 95, 135

- W -

Warning Points 251
Warnings 272
Web OpenType Font Format 285
Weight 211
Width 211
Win Ascent 156, 217
Win Descent 156, 217
WOFF 285
WOFF2 285
WWS 211

- X -

x-Height 156, 217

- Z -

Zoom 56