

Handbook July 2016



Glyphs 2.3

Create – Produce – Release
Font Editing for Everyone

You are reading the Glyphs Handbook from July 2016 for the application version 2.3. Please download the latest version of this handbook at:

glyphsapp.com/get-started

© 2011–2016 glyphsapp.com

Written by Rainer Erich Scheichelbauer and Georg Seifert.

Thanks to Jeff Kellem, Rob Keller, Toshi Omagari and Claus Eggers Sørensen for their invaluable input.

Contents

- 1 Glyphs 8**
 - 1.1 A Tool for Creating OpenType Fonts 8
 - 1.2 Installation 8
 - 1.3 Community 9
 - 1.4 Updates 9
 - 1.5 Glyphs Mini 9
 - 1.6 Keyboard Shortcuts 9
 - 1.7 Crash Reports 10

- 2 Preferences 11**
 - 2.1 Accessing Application Preferences 11
 - 2.1.1 Updates 11
 - 2.1.2 User Settings 12
 - 2.1.3 Sample Strings 13
 - 2.1.4 Sharing 14
 - 2.1.5 Add-ons 14

- 3 Edit View 16**
 - 3.1 Editing Glyphs 16
 - 3.2 Drawing Paths 16
 - 3.2.1 Draw Tool 16
 - 3.2.2 Pencil Tool 17
 - 3.2.3 Primitives 17
 - 3.3 Editing Paths 18
 - 3.3.1 Selecting Nodes and Paths 18
 - 3.3.2 Moving Selected Nodes and Paths 18
 - 3.3.3 Converting Nodes and Segments 19
 - 3.3.4 Nodes in Alignment Zones 19
 - 3.3.5 Scaling and Rotating 20
 - 3.3.6 Aligning 21
 - 3.3.7 Duplicating Paths 21
 - 3.3.8 Deleting Nodes 22
 - 3.3.9 Opening and Closing Paths 23
 - 3.3.10 Cutting Paths 23
 - 3.3.11 Resegmenting Outlines 24
 - 3.3.12 Controlling Path Direction 25
 - 3.3.13 Extremes and Inflections 26
 - 3.3.14 Duplicate Nodes 27
 - 3.4 Anchors 27
 - 3.4.1 Compound and Positioning Anchors 27
 - 3.4.2 Ligature Carets 27
 - 3.4.3 Adding, Editing and Removing Anchors 28
 - 3.4.4 Mark to Base Positioning 29
 - 3.4.5 Mark to Mark Positioning 29
 - 3.4.6 Cursive Attachment Anchors 30
 - 3.5 Guidelines 30
 - 3.5.1 Magnetic Guidelines 30
 - 3.5.2 Local and Global Guidelines 30
 - 3.5.3 Glyph-Specific Undo History 31
 - 3.6 Glyph Display 32
 - 3.6.1 Zooming 32
 - 3.6.2 Panning 33
 - 3.6.3 View Options 33
 - 3.6.4 Glyph and Layer Colors 33
 - 3.7 Background 34
 - 3.8 Entering Text 35
 - 3.8.1 Sample Texts 36
 - 3.8.2 Text Tool 37
 - 3.8.3 Writing Direction 38
 - 3.9 Measuring 38
 - 3.9.1 Info box 38
 - 3.9.2 Measurement Tool 40
 - 3.9.3 Measurement Guidelines 41
 - 3.9.4 Measurement Line 42
 - 3.10 Annotating 43
 - 3.10.1 Annotation Cursor 43
 - 3.10.2 Annotation Text 43
 - 3.10.3 Annotation Arrow 44

- 3.10.4 Annotation Circle 44
- 3.10.5 Plus and Minus Annotations 44
- 3.11 Images 44
 - 3.11.1 Adding Images 44
 - 3.11.2 Manipulating Images 45
- 3.12 Previewing and Testing 46
 - 3.12.1 Previewing Kerning 46
 - 3.12.2 Previewing Masters 46
 - 3.12.3 Previewing Path Offset 46
 - 3.12.4 Previewing OpenType Features 47
 - 3.12.5 Previewing Interpolated Instances 47
 - 3.12.6 Previewing in OS X 48
 - 3.12.7 Previewing in Adobe Applications 49
 - 3.12.8 Previewing in Web Browsers 50
- 4 Palette 51**
 - 4.1 Palette Sidebar 51
 - 4.2 Dimensions 51
 - 4.3 Fit Curve 51
 - 4.4 Layers 52
 - 4.4.1 Working with Layers 52
 - 4.4.2 Special Layers 53
 - 4.5 Transformations 53
 - 4.5.1 Transformation Origin 53
 - 4.5.2 Mirroring 54
 - 4.5.3 Reversible Transformations 54
 - 4.5.4 Aligning 54
 - 4.5.5 Boolean Operations 55
- 5 Filters 56**
 - 5.1 Filters 56
 - 5.1.1 Filters Menu 56
 - 5.1.2 Filters as Custom Parameters 56
 - 5.2 Built-In Filters 57
 - 5.2.1 Fix Compatibility 57
 - 5.2.2 Hatch Outline 57
 - 5.2.3 Offset Curve 58
 - 5.2.4 Remove Overlap 59
 - 5.2.5 Roughen 60
 - 5.2.6 Round Corners 60
 - 5.2.7 Rounded Font 60
 - 5.2.8 Transformations 61
 - 5.3 Third-Party Filters 62
- 6 Font View 64**
 - 6.1 Viewing Glyphs 64
 - 6.1.1 Grid View 64
 - 6.1.2 List View 65
 - 6.1.3 Searching for Glyphs 66
 - 6.2 Managing the Glyph Set 66
 - 6.2.1 Generating New Glyphs 66
 - 6.2.2 Copying Glyphs Between Files 68
 - 6.2.3 Removing Glyphs 69
 - 6.3 Glyph Properties 70
 - 6.3.1 Name 70
 - 6.3.2 Width and Sidebearings 70
 - 6.3.3 Kerning Groups 70
 - 6.3.4 Exports 71
 - 6.3.5 Color Label 71
 - 6.3.6 Unicode 71
 - 6.3.7 Note (List View) 72
 - 6.3.8 Components (List View) 72
 - 6.3.9 Read-Only Properties in List View 72
 - 6.4 Batch-Processing 72
 - 6.4.1 Selecting Glyphs 72
 - 6.4.2 Menu Commands 73
 - 6.4.3 Batch-Renaming Glyphs 73
 - 6.4.4 Filters 74
 - 6.4.5 Palette Manipulations 74
 - 6.5 Filtering and Sorting 74
 - 6.5.1 Search Box 74
 - 6.5.2 Categories 75
 - 6.5.3 Languages 76
 - 6.5.4 Smart Filters 76
 - 6.5.5 List Filters 77
 - 6.5.6 Manage Filters 77

- 6.5.7 Custom Parameter ‘glyphOrder’ 78
- 6.6 Names and Unicode 78
 - 6.6.1 Glyph Info Database 78
 - 6.6.2 Naming Glyphs 80
 - 6.6.3 Renaming Glyphs 82
 - 6.6.4 CID Mapping 82
- 6.7 Images 82
 - 6.7.1 Adding and Managing Images 82
 - 6.7.2 Viewing Images 83

7 Font Info 84

- 7.1 Font 84
 - 7.1.1 Family Name 84
 - 7.1.2 Units per Em 84
 - 7.1.3 Designer and Designer URL 85
 - 7.1.4 Manufacturer and Manufacturer URL 85
 - 7.1.5 Copyright 85
 - 7.1.6 Version 85
 - 7.1.7 Date 86
 - 7.1.8 Custom Parameters 86
- 7.2 Masters 86
 - 7.2.1 Proportions: Weight, Width, and Custom 87
 - 7.2.2 Metrics 87
 - 7.2.3 Stems 88
 - 7.2.4 Alignment Zones 89
 - 7.2.5 Custom Parameters 89
- 7.3 Instances 90
 - 7.3.1 Is Active 90
 - 7.3.2 Style Name 90
 - 7.3.3 Weight and Width 91
 - 7.3.4 Style Linking 92
 - 7.3.5 Interpolation 93
 - 7.3.6 Custom Parameters 93
 - 7.3.7 Instance Preview 93
- 7.4 Features 93
 - 7.4.1 OpenType Feature Code 93
 - 7.4.2 Automatic Feature Code 94
 - 7.4.3 Manual Feature Code 94

- 7.5 Other Settings 96
 - 7.5.1 Grid Spacing and Subdivision 96
 - 7.5.2 Use Custom Naming 96
 - 7.5.3 Disable Automatic Alignment 97
 - 7.5.4 Keep Alternates Next to Base Glyph 97
- 7.6 Notes 97
 - 7.6.1 Font Note 97

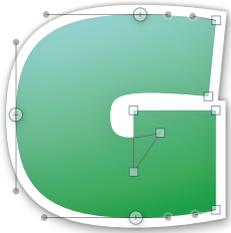
8 Reusing Shapes 98

- 8.1 Components 98
 - 8.1.1 Building Compounds 98
 - 8.1.2 Turning Paths into Components 99
 - 8.1.3 Recipes 99
 - 8.1.4 Editing Components 100
 - 8.1.5 Moving between Base Glyphs and Compounds 100
 - 8.1.6 Anchors 101
 - 8.1.7 Automatic Alignment 103
 - 8.1.8 Locking Components 104
 - 8.1.9 Decomposing 105
 - 8.1.10 Combining Paths and Components 105
 - 8.1.11 Nesting Components 105
 - 8.1.12 Preferred Marks for Glyph Composition 106
 - 8.1.13 Underscore Components 106
- 8.2 Smart Components 106
 - 8.2.1 Setting up Smart Glyphs 107
 - 8.2.2 Adding Smart Components 108
 - 8.2.3 Smart Component Settings 109
 - 8.2.4 Part Anchors 109
- 8.3 Corner and Cap Components 109
 - 8.3.1 Corner Components 109
 - 8.3.2 Caps 111

- 9 Spacing and Kerning 114**
 - 9.1 Spacing 114
 - 9.1.1 Spacing Shortcuts 114
 - 9.1.2 Metric Keys 114
 - 9.1.3 Metric Keys and Automatic Alignment 116
 - 9.2 Kerning 116
 - 9.2.1 Ways to Kern 116
 - 9.2.2 Kerning Groups 117
 - 9.2.3 Finding and Viewing Kerning Pairs 118
 - 9.2.4 Deleting Kerning Pairs 119
 - 9.2.5 Copying Kerning Pairs 119
 - 9.2.6 Cleaning up Kerning 119
 - 9.2.7 Compressing Kerning 120
 - 9.2.8 Optional 'kern' Feature Lookup 120
- 10 PostScript Hinting 121**
 - 10.1 Hinting 121
 - 10.2 Font-Wide Hints 122
 - 10.2.1 Standard Stems 122
 - 10.2.2 Alignment Zones 123
 - 10.2.3 Custom Parameters 125
 - 10.3 Autohinting 125
 - 10.3.1 Flex Hints 125
 - 10.4 Manual hinting 126
 - 10.4.1 Stem Hints 127
 - 10.4.2 Ghost Hints 129
 - 10.4.3 Hinting Multiple Masters 130
- 11 TrueType Hinting 132**
 - 11.1 Instructioning 132
 - 11.2 Autohint 132
 - 11.2.1 Either Manual or Automatic Instruction 132
 - 11.3 Manual Instructions 133
 - 11.3.1 Horizontal Stems and Zones 134
 - 11.3.2 Rasterizer Preview 135
 - 11.3.3 Anchor 136
 - 11.3.4 Align 136
 - 11.3.5 Stem 137
 - 11.3.6 Triple Hint 139
 - 11.3.7 Interpolate 139
 - 11.3.8 Diagonal 140
 - 11.3.9 Automatic Interpolation of Untouched Points 140
- 12 Multiple Masters 141**
 - 12.1 Overview 141
 - 12.2 Setting up Masters 142
 - 12.3 Setting up Instances 143
 - 12.4 Fix Outline Incompatibility 144
 - 12.5 Comparing Master Layers 147
 - 12.6 Ensuring Family Consistency across Files 148
 - 12.7 Brace Layers 149
 - 12.8 Bracket Layers 149
 - 12.9 Open Bracket Layers 150
- 13 Color Fonts 151**
 - 13.1 Working on Multiple Layers 151
 - 13.1.1 Select All Layers Tool 151
 - 13.1.2 Keeping the Metrics in Sync 151
 - 13.1.3 Exporting Color Fonts 151
 - 13.2 Layered Color Fonts 152
 - 13.2.1 Named Masters with Master Colors 152
 - 13.2.2 Instances for All Masters 152
 - 13.2.3 Previewing and Using Layered Color Fonts 152
 - 13.3 Microsoft Color Fonts 153
 - 13.3.1 CPAL Table 153
 - 13.3.2 Creating Color Layers 154
 - 13.4 Apple Color Fonts 155
 - 13.4.1 Bitmap Image Files 155
 - 13.4.2 Resolution Layers 156
 - 13.4.3 Using Apple Color Fonts 156
 - 13.5 SVG Color Fonts 156

13.5.1	SVG Image Files	157	15.4.2	Generating OpenType/CFF Fonts	165
13.5.2	SVG Layers	157	15.4.3	Generating OpenType/TT Fonts	166
13.5.3	Using SVG Color Fonts	157	15.5	Webfonts	167
14	Error Handling	158	15.5.1	Generating WOFF, WOFF2, EOT	167
14.1	Glyph Names	158	15.6	Metrics	168
14.2	Font Names	158	15.6.1	Import Metric Data	168
14.3	Duplicate Unicode Values	159	15.6.2	Export Metrics	169
14.4	OpenType Feature Code	159	15.7	Projects	169
14.5	Missing Outlines	162	15.7.1	Setting up a Project	170
14.5.1	Open Paths	162	15.8	Exporting a Project	171
14.5.2	Wrong Path Orientation	162	16	Extensions	172
14.5.3	Multiple Paths on Top of Each Other	162	16.1	Scripts	172
14.5.4	Outline Incompatibility	162	16.2	Plugins	172
15	Import and Export	163	16.2.1	Manual Plugin Installation	172
15.1	Vector Drawing Applications	163	16.2.2	Plugin Manager	173
15.1.1	Adobe Illustrator	163	16.3	SDK	174
15.1.2	Sketch	163	17	Appendix	175
15.1.3	Copy and Paste the Paths	163	17.1	Automatic Feature Generation	175
15.2	FontLab Studio	164	17.2	Automatic Class Generation	179
15.2.1	From FontLab Studio 5 to Glyphs	164	17.3	Custom Parameters	180
15.2.2	From Glyphs to FontLab Studio 5	164	17.4	Changing the Glyph Data	204
15.3	Unified Font Object	164	17.4.1	Global Glyph Data Changes	204
15.3.1	Native Saving Format	164	17.4.2	Local Glyph Data Changes	206
15.3.2	Exporting to UFO	164	17.5	Changing Tool Shortcuts	206
15.3.3	Importing UFO Files	164			
15.4	Type 1, OpenType, and TrueType	165			
15.4.1	Opening Existing Fonts	165			

1 Glyphs



1.1 A TOOL FOR CREATING OPENTYPE FONTS

Glyphs is primarily a tool for designing and producing new fonts. Its main principle is that you can edit glyphs in a word context. All tools are optimized for a type design workflow as natural, quick, and intuitive as possible.

We believe that you should be able to focus on your design and only be bothered with technicalities if it is really necessary. There is no need for keeping a design version next to a production version of your font. All production steps take place at export time. By default, Glyphs automates a lot of technical details for you, but you can always override automation manually.

Glyphs exports OpenType fonts of all flavors: OpenType/CFE (Compact Font Format, suffix .otf), OpenType/TT (TrueType, suffix .ttf), and the webfont formats WOFF and WOFF2 (Web Open Font Format, suffixes .woff and .woff2), and EOT (Embeddable OpenType, suffix .eot). All modern software, including all modern web browsers, supports OpenType fonts. However, outdated legacy software may require pre-OpenType and pre-Unicode TTFs or PostScript Type 1 fonts, which cannot be produced with Glyphs.

Glyphs can open existing fonts. However, in the import process, some information stored in the font may be lost. For details, see chapter 15, 'Import and Export' (p. 163).

1.2 INSTALLATION

Glyphs is a Mac-only application. Glyphs 2.0 or higher requires at least OS X 10.9.5 Mavericks. Download the trial from glyphsapp.com/buy and move the app into your Applications folder, which you can access in Finder via *Go > Applications* (Cmd-Shift-A). To unlock the trial, open the .glyphszlicense file you purchased, either by simply double clicking the license file, or by dragging it onto the app icon.

1.3 COMMUNITY

If you have questions or suggestions, you can register and post in the Glyphs forum at forum.glyphsapp.com. Because of spam protection requirements, you must be a registered Twitter user, and your first posting must be approved by one of the moderators. For bug reports, you can register at bugreport.glyphsapp.com.

1.4 UPDATES

The application is updated on a regular basis. The latest feature additions are described on glyphsapp.com/blog, and a detailed change log is accessible through *Help > Change Log*. You can check which version you have by choosing *Glyphs > About Glyphs* and trigger an update with *Glyphs > Check for Updates* or by activating automatic updates in *Glyphs > Preferences > Updates*. For more details, see section 2.1.1, ‘Updates’ (p. 11).



1.5 GLYPHS MINI

Glyphs Mini is a trimmed-down, light version of the application. It lacks many of the advanced features of the full application described in this handbook, e.g., support for plug-ins, glyph layers, Multiple Masters, Python scripting, custom OpenType feature code, custom parameters, corner, cap and smart components, and manual hinting. It is intended as a simple and affordable entry-level solution for casual type design or putting together a dingbat (symbol) font.

1.6 KEYBOARD SHORTCUTS

To enable a workflow as efficient as possible, Glyphs employs a number of keyboard shortcuts. You can set your own shortcuts in the OS X System Preferences, under *Keyboard > Shortcuts > App Shortcuts > Glyphs*.

Some shortcuts conflict with default shortcuts for system functions like Cmd-Space for Spotlight or Ctrl-Arrows for switching between Spaces. To enable the full functionality of the application, it is advisable to change these system shortcuts in the System Preferences.

1.7 CRASH REPORTS

After an application crash, you will see two crash report dialogs. First, a generic one that goes to Apple and never reaches us. Secondly, when the app restarts, you will see a report dialog that sports the Glyphs app icon in the top left. This second report is essential for us to fix the problem that caused the crash. So, please, in your own interest, *always* send those reports. If you choose to send a crash report after an application crash, we will receive some technical information about your hardware, your OS, and the last state of the Glyphs application, and some additional information such as the exact time your crash happened.

When run for the first time, Glyphs will ask for access to your contacts, in order to auto-fill your name and e-mail address in the crash reports. If you do not wish to grant that access, you can manually enter your info when the crash report dialog appears, or simply leave the fields blank.

Providing your contact information, however, allows us to get back to you. In order to fix the bug that caused the crash on your machine, we may ask you for the file you were working on when the crash happened. We delete the files after we have dealt with your problem, and do not hand them over to a third party without your explicit consent.

If you can reliably reproduce a crash, it is a good idea to describe the steps for reproducing it, either directly in the crash report dialog, in the forum, or in a bug report on bugreport.glyphsapp.com.



Problem Report for Glyphs

 Glyphs unexpectedly quit. Would you like to send a report so we can fix the problem?

Name Email

▼ Comments

Only the presented data will be sent with this report.

2 Preferences

2.1 ACCESSING APPLICATION PREFERENCES

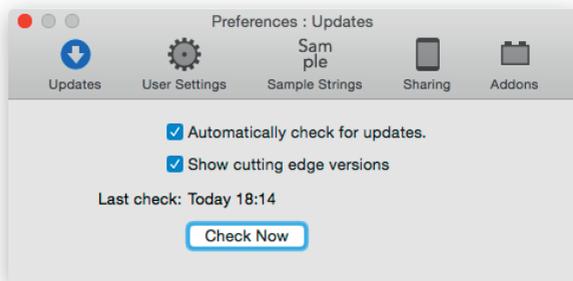
Open the Preferences window through *Glyphs > Preferences* (Cmd-comma) to access all application-wide preferences.

2.1.1 Updates

The *Updates* section controls how Glyphs handles software updates. Clicking on the *Check now* button will initiate a version check and, if available, offer you a new version for download and installation, presenting you a change log describing the latest additions and changes. The time stamp of the latest check is displayed next to the button.

If enabled, the *Automatically Check for Updates* option will periodically try to establish an internet connection and see if a new app version is available for download from the Glyphs servers. We recommend keeping this option on.

Show Cutting Edge Versions will also offer you beta versions, rather than only stable versions. Glyphs beta versions are frequently released, and bugfixes are available much sooner this way. Enable this version if you cannot wait for the stable version or want to experiment with new app features. Because of the high release frequency, beta versions are not as thoroughly tested as stable versions. That is why we recommend to always work with file copies if you are testing cutting edge versions.



You can revert back to the stable version by downloading the trial from glyphsapp.com/buy.

2.1.2 User Settings

The option *Keep Glyph Names from Imported Files* will not force the internal glyph naming scheme onto files originating from other applications. This is useful if you want to employ a workflow where you need to rely on a certain naming scheme, e.g., if you exchange data with other apps.

When enabled, *Disable Automatic Alignment for Imported Files* will disable the automatic positioning for all components in fonts opened from other applications. For more details on automatic alignment, see section 8.1.7, 'Automatic Alignment' (p. 103).

Text View Width controls the line width of text entered in Edit view. The value entered is measured in thousandths of an em. This measurement is independent from your *Units per Em* value as entered in Font Info.

Handle Size controls the size of all displayed points in Edit view, such as on-curve nodes, control handles, anchors, and position markers of manual hints. Larger nodes make it easier to click-select a point, while smaller nodes make for a cleaner user interface.

You can set the display colors of corner nodes, smooth nodes, alignment zones, positive and negative kerning indicators in the Edit view, and the canvas color. The *Standard* button resets the color values to their defaults.

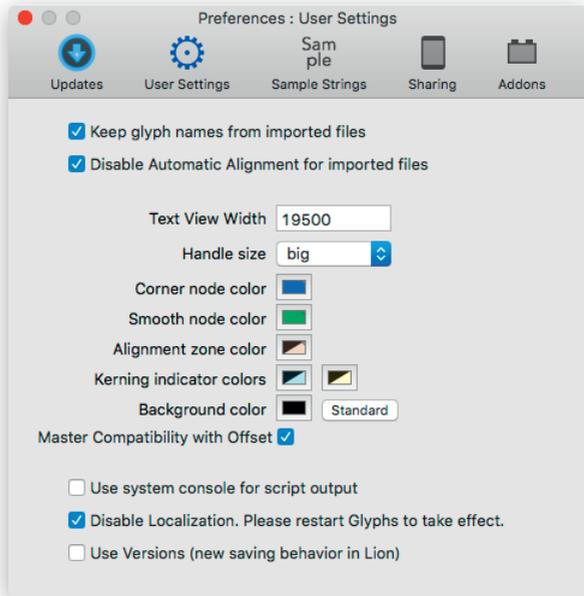
Master compatibility with offset sets the way Glyphs compares outlines when *View > Show Master Compatibility* (Ctrl-Opt-Command-N) is on. Glyph masters are displayed on top of each other if the option is disabled, or in an exploded view if it is enabled. For more details, see section 12.4, 'Fix Outline Incompatibility' (p. 144).

The option *Use system console for script output* will redirect the standard out of the scripting API to Console.app, rather than the Macro Panel (*Window > Macro Panel*). This is useful if a script crashes Glyphs and you cannot see its output otherwise.

Disable Localization will keep the interface language of Glyphs in English rather than the localization you have set in your System Preferences.

Use Versions will make Glyphs employ the system's saving method, 'Versions', introduced in OS X 10.7 Lion. It automates file saving, and enables version browsing. However, you may prefer to take file saving in your own hands, especially if you

are keeping your files in cloud services that utilize their own version control mechanisms that, due to the very frequent incremental saves, may become hard or impossible to use if *Use Versions* is on.



2.1.3 Sample Strings

Here, you can import, edit, reset, copy, or paste default sample strings, separated by newlines. The text field is Unicode savvy, so you can enter any diacritic and non-Latin characters. Use a backward slash followed by a lowercase n (\n) for a newline in the sample string. Use a forward slash followed by the glyph name and a separator character (either a space or another forward slash) as an escape for any glyph. This is especially useful for hard-to-type or unencoded glyphs, e.g., OpenType variations such as '/a.sc'. Use '/Placeholder' (with a forward slash and a separator character at the end) for the Placeholder function, i.e., the current glyph in Edit view.

The *Open File* button allows you to import an external text file, whereas the *Standard* button will reset the sample texts to the app defaults. For more details on how to use sample strings, (see 3.8.1, 'Sample Texts', p. 36).

2.1.4 Sharing

With the *Enable External Preview* option, Glyphs can mirror the content of the current Edit tab on an iOS device, such as an iPhone, iPod Touch, or iPad. If you want to make use of this functionality, install the free 'Glyphs Viewer' app from the iOS AppStore. The iOS device and the Mac running Glyphs must be on the same network. Start the Glyphs Viewer app and select the app which you want to mirror on your handheld device. Tap and hold anywhere on the screen to go back to the menu.

If Glyphs Viewer cannot find your machine, make sure both the Mac and the iOS device are connected to the same wireless network. Restart both machines and try again, in case the problem persists. If it still cannot connect, try in a different network, or adjust your router settings. If Mac and iOS device communicate to the router with different wireless standards, Glyphs Viewer will not be able to connect to the Mac. This can happen if the router is set up to use multiple modulation standards (e.g., g and n) simultaneously. Setting the router to either 802.11g only or 802.11n only may help in such cases.

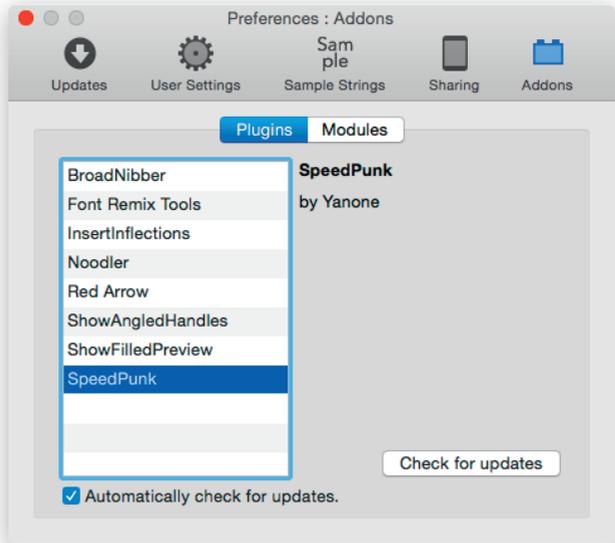
2.1.5 Add-ons

All installed plugins are listed in the *Plugins* tab. If the developer embedded the necessary information inside the plugin, Glyphs can also notify you of plugin updates. Depending on the embedded information provided by the developer, an option for downloading the new version, or for taking you to the plugin website can be displayed.

If the option *Automatically check for updates* is enabled, Glyphs will notify you of updates when the app has started up, and offer to take you to this tab in the *Preferences* window. Plugins for which an update is available, will be marked with an asterisk. Selecting the plugin will bring up a *Download* button in the bottom right of the window. A click on the button will take you to the plugin homepage as indicated by its author.

To install a plugin, simply open the plugin file in Finder, and confirm the dialog that appears in Glyphs. After an app restart, the plugin's functionality is available. To remove an installed plugin, right-click on the name of the plugin in the list, and choose *Show in Finder* from the context menu. Move the plugin file to the Trash or anywhere out of the

Plugins folder, and restart the application for the changes to take effect.



In the *Modules* tab, the *Install Modules* button will download and install the third-party open-source Python libraries Vanilla by Tal Leming, RoboFab by Tal Leming, Erik van Blokland and Just van Rossum, and FontTools by Just van Rossum as maintained by Behdad Esfahbod. Some scripts available for Glyphs make use of these libraries. If you intend to install third-party scripts, or write your own Python scripts for extending Glyphs, we strongly recommend installing these libraries. Also, a Python wrapper for RoboFab called `objectsGS.py` will be placed in your Scripts folder. RoboFab scripts work with their usual imports at the beginning, e.g., `from robofab.world import CurrentFont`.

3 Edit View

3.1 EDITING GLYPHS

In Edit view, you can edit glyphs. To access Edit view, you need to open an Edit tab with *View > New Tab* (Cmd-T). To access an existing tab, either click on the respective tab title or press the number of the tab (2 through 9) while holding down Cmd-Opt. The first tab (Cmd-Opt-1) is always the Font view. For more details, see chapter 6, 'Font View' (p. 64).

The Edit view has two modes, edit mode and text mode. Access text mode by activating the Text tool (keyboard shortcut T). For more details, see section 3.8, 'Entering Text' (p. 35). Switch into edit mode by either choosing one of the other tools, double clicking one of the displayed glyphs, or pressing the Esc key when the text cursor is situated in front of a glyph, i.e., to the left when in left-to-right writing mode, to the right when in right-to-left mode, and on top when in top-to-bottom mode.

3.2 DRAWING PATHS

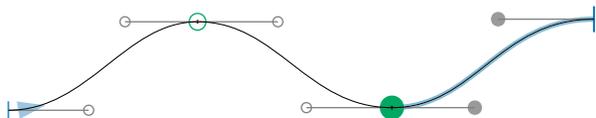
Create new paths with the Draw, Pencil or Primitives tools.

3.2.1 Draw Tool

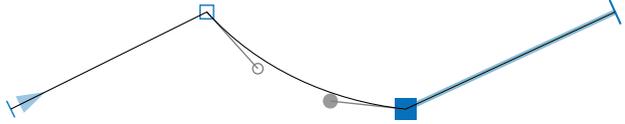
When the Draw tool  (shortcut P as in 'Path' or 'Pen') is active, click anywhere in the canvas to create straight lines, or click and drag to create curve segments. If you keep your mouse button pressed, you can move the on-curve point by holding down the space bar on your keyboard. Close the outline by clicking on its first node.

Nodes in smooth connections will appear as green circles. You can set a different color in *Glyphs > Preferences > User Settings*. A smooth connection can either be a curve (an on-curve point in line with two surrounding off-curve points) or a tangent (an on-curve point in line with another on-curve point and an off-curve point). You can trigger the display of nodes with *View > Show Nodes*.

This handbook uses the term 'node' to refer to on-curves, 'handle' for off-curves, and 'point' as an umbrella word for both types.



In order to draw a corner, hold down the Option key while dragging. Corner points are marked by blue squares. Again, you can choose a different color in *Glyphs > Preferences > User Settings*. Points in a corner connection are not kept in line, so you can move them independently from each other.



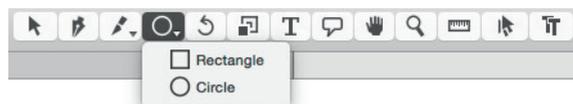
Handles (Bézier control points, off-curve points) control the curvature of the path segment and are displayed as small gray circles. In open paths, start and end points are displayed as short perpendicular blue lines. The start point also features a blue triangle, indicating the path direction, i.e., the stored order of the points. The path direction is important for certain path functions, glyph rendering (see 3.3.12, 'Controlling Path Direction', p. 25), and corner and cap components (see 8, 'Reusing Shapes', p. 98).

3.2.2 Pencil Tool

The Pencil tool  (shortcut B) offers a quick way to draw freehand curves, especially for people using tablets. The resulting paths will need some cleaning up. That is because the Pencil paths will usually contain too many nodes in order to stay as close as possible to the cursor movement.

3.2.3 Primitives

Glyphs offers rectangles and ovals as built-in primitive shapes. Click the tool or press F to activate it in its current mode. Click and hold the Primitives tool or press Shift-F to choose between the two shape options. Alternatively, you can use *Draw Circle* or *Draw Rect* from the context menu to switch between the two functions.



Click once on the canvas to create a primitive by entering its measurements with the keyboard. Or click and drag to draw it directly into the edit area. Hold down Shift for a perfect square or circle. Hold down Option to draw from the center of the shape.

3.3 EDITING PATHS

3.3.1 Selecting Nodes and Paths

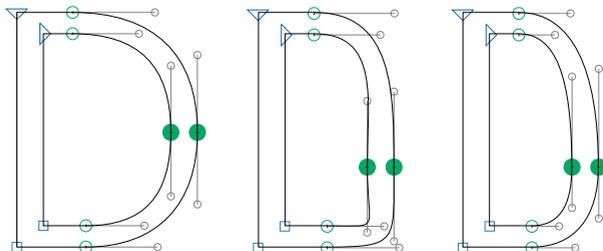
Click and drag with the Select tool  (shortcut V) to select nodes and handles inside a rectangular selection area. Hold down the Option key to ignore the handles and only select on-curve nodes. Double click on or near an outline segment to select complete paths. Hold down the Shift key to extend or reduce the selection.

Glyphs allows you to select multiple handles independently of the nodes. You can also Shift-select any number of handles in a non-contiguous selection. When a single node or handle is selected, press the Tab key to select the following point on the path, or Shift-Tab to go to the previous one.

3.3.2 Moving Selected Nodes and Paths

Move a selection using the mouse or the cursor keys. Moving nodes will move the attached handles even if they are not selected. Hold down the Shift key for increments of 10, and the Cmd key for increments of 100. Hold down the Option key to move only explicitly selected on-curve points. While moving one or more nodes, hold down both Ctrl and Option (or add Option after you started dragging) to 'nudge' them, i.e., to proportionally adjust the surrounding unselected handles at the same time.

Left: original glyph outline with two selected nodes.
Center: selected nodes moved, handles stay the same.
Right: selected nodes nudged, handles are adjusted.



To move a handle, simply drag it with your mouse. Or select it and use your arrow keys. If more than one handle is selected, you can move them all simultaneously. Moving one or more handles while holding down the Option key preserves their angles. Again, add the Shift key for increments of 10, or the Cmd key for increments of 100.

While moving a handle next to a smooth (green) node, you can hold down Ctrl and Option simultaneously to mirror the length and angle of the adjacent handle on the other side of the node.

Click anywhere on the segment and drag to quickly change the shape of the segment. Option-drag a segment to change the handle lengths, but keep their respective angles.

3.3.3 Converting Nodes and Segments

Convert between (green) smooth connections and (blue) corners by double clicking an on-curve point, or by selecting one or more nodes and pressing Return.

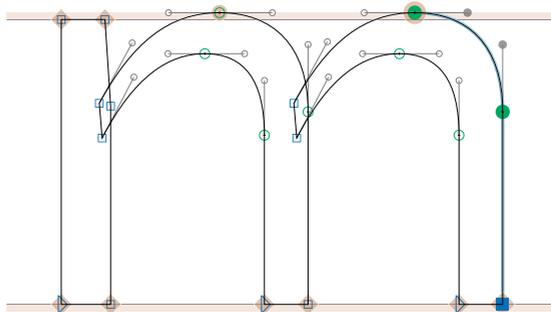
The *Glyph > Tidy up Paths* command (Cmd-Opt-Shift-T) applies heuristics to set the appropriate mode for all nodes at once, or all selected nodes if there is a selection. It also removes superfluous points, e.g., handles on a straight segment, or an on-curve point exactly in line between two others.

Option-clicking a line segment converts it into a curve segment, i.e., adds handles. To convert a curve back into a line segment, select and delete one or both of its handles.

Be careful when tidying up paths: In Multiple Master setups, superfluous points may be necessary for outline compatibility.

Tip: Quickly add handles by Option-clicking on the outline between two nodes.

3.3.4 Nodes in Alignment Zones



Nodes located exactly on a vertical metric line (see 7.2, 'Masters', p.86) are highlighted with a beige diamond. Inside

an alignment zone, the highlighting assumes the shape of a circle. This helps controlling the position of nodes even at small zoom scales.

3.3.5 Scaling and Rotating

The attributes of the current node selection are shown in the gray Info box (*View > Show Info, Cmd-Shift-I*):

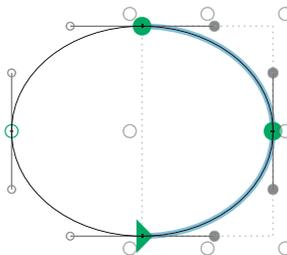


Tip: In all number input fields throughout the application, you can use the up and down cursor keys to increase or decrease the value. Simultaneously holding down the Shift key gives you increments of 10.

When more than one node is selected, you can scale or move the selection by changing the numbers for its position (x and y) and its dimensions (↔ for width, ⇅ for height) in the Info box. Set the transformation origin with the nine reference points on the left. Close the lock symbol  to scale width and height proportionally. Open the lock  to distort the selection, i.e., scale width and height independently from each other. You can use the up and down arrow keys to step through the numbers. Hold down Shift for increments of 10.

When multiple nodes are selected, the number next to the solid square will indicate how many nodes and handles are currently selected. The number next to the outlined square represents the total number of on-curve and off-curve points on the current glyph layer.

You can also rotate and scale your selection manually with the Rotate tool  (shortcut R) and the Scale tool  (shortcut S). With one of these tools, click anywhere on the canvas to set the transformation origin, then click and drag to transform the current selection. Hold down the Shift key to rotate in steps of ninety degrees, or scale proportionally.



When more than one node is selected, you can choose to display a bounding box for the selection with *View > Show Bounding Box* (Cmd-Opt-Shift-B). Drag any of the white knobs to scale with the opposite end of the selection as transformation origin. Hold down the Shift key to scale proportionally, and the Option key to use the center as transformation origin.

More path transformations are possible via the Palette. See section 4.5, ‘Transformations’ (p.53) for further details.

3.3.6 Aligning

Choose *Paths > Align Selection* (Cmd-Shift-A) to quickly align all selected points. The command aligns both nodes and handles. Glyphs will then automatically choose between horizontal and vertical alignment, whichever is smaller for the current selection. The *Align Selection* command respects the transformation origin of the gray Info box. Alternatively, you can either set the width or the height value of two or more selected points to zero in the Info Box.

You can center an anchor horizontally between to points if you run the *Align Selection* command while two points and one anchor are selected.

Running the *Align Selection* command while exactly one point and one component are selected, will align the origin point of the component to the selected node. The node keeps its position. The origin point is where the baseline crosses the left sidebearing when the italic angle is zero. If the component contains an anchor called ‘origin’, Glyphs will use the position of that anchor instead of the origin point for aligning the component to a node.

Applying *Paths > Align Selection* on a single node will try to move the node over the nearest node in the background.

Paths > Align Selection works on individual nodes. To also align partial paths, complete paths or components to each other, use the *Transformations* section of the Palette sidebar (Opt-Cmd-P). See chapter 4.5, ‘Transformations’ (p.53), for more details.

3.3.7 Duplicating Paths

To quickly duplicate a complete path, first select it entirely by double clicking on or near it. Repeat with the Shift key to add or subtract paths to the selection. Then, hold down the Option

If the italic angle is not zero, instead of the left sidebearing, an imaginary vertical line that crosses the slanted LSB at half x-height is used. In that case, the origin point is where this line crosses the baseline.

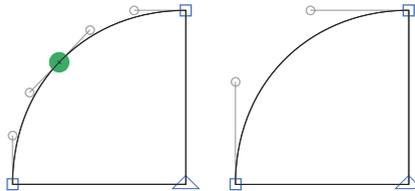
key while you click and drag a copy of the paths into their new position.

Alternatively, you can simply copy (Cmd-C) and paste (Cmd-V) the selected artwork. This method may make more sense if you plan to position the new paths with the keyboard rather than with the mouse.

Option-dragging partial paths will duplicate the selected segments. This can be helpful when replicating glyph parts like serifs or spurs.

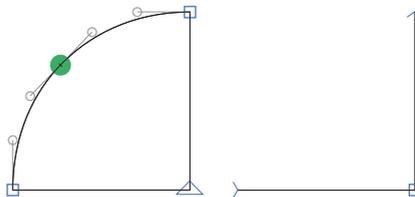
3.3.8 Deleting Nodes

Simply select a node and press the Delete key to delete the node. Alternatively, you can use the Erase tool  (shortcut E). Glyphs will keep the path closed and try to reconstruct the path segment without the node:



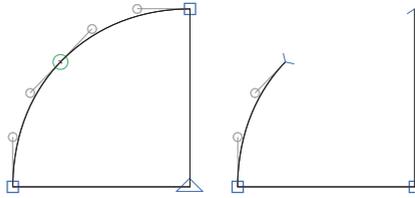
Tip: A quick way to get rid of a path segment between two nodes is to insert a point with the Draw tool (shortcut P) and immediately Option-delete it.

Hold down the Option key to break the path, i.e., remove the node and both path segments surrounding the node:



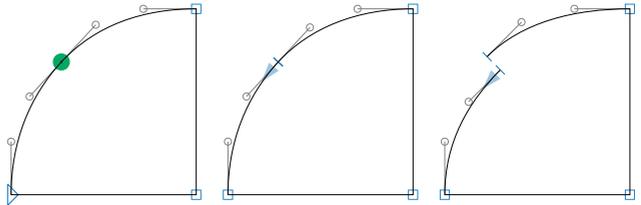
To get rid of a segment between two on-curve nodes, switch to the Erase tool  (shortcut E or Shift-E), and while you hold down the Option key, click on the path segment. Alternatively, you can select a handle and press Opt-Delete. Or select the complete segment, including the on-curve nodes at the edges,

and press Opt-Delete. This also works for multiple segments, adjacent or non-adjacent:



3.3.9 Opening and Closing Paths

With the Draw tool  (shortcut P), click on a node to open the path in the position of the node. Open path endings are marked by short blue perpendicular lines. You can now drag them apart with the Select tool  (shortcut V).



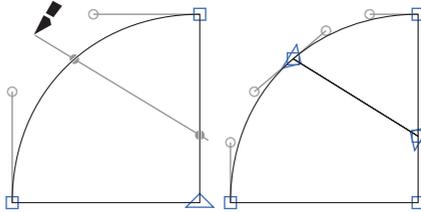
To close the path again, simply drag an open line ending on top of another with the Select tool (shortcut V). Alternatively, you can close all open paths in a glyph by right-clicking anywhere in the canvas and choosing *Close Open Paths* from the context menu. Or select two open ends, and choose *Connect Nodes* from the context menu.

3.3.10 Cutting Paths

With the Knife tool  (shortcut E or Shift-E), click and drag a line across a path, to cut the outline into two separate outlines. Glyphs will close the two resulting paths along the

Tip: When multiple tools share one icon in the toolbar such as the Knife and Erase tools, you can add Shift to the tool shortcut to toggle between the tools.

cutting line. If you cut across several overlapping paths, it will rewire the segments with each other.

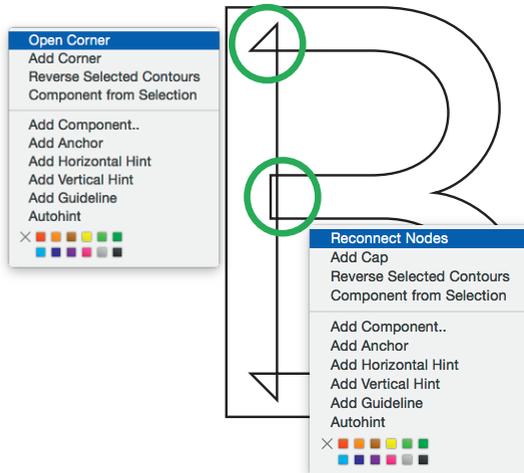


To activate the Knife tool when it is not displayed in the toolbar, click and hold the Erase tool , and choose Knife from the pop-up menu. Alternatively, you can press Shift-E.



3.3.11 Resegmenting Outlines

The results of the Open Corner and Reconnect Nodes operations. Opening corners only works on (blue) corner points.

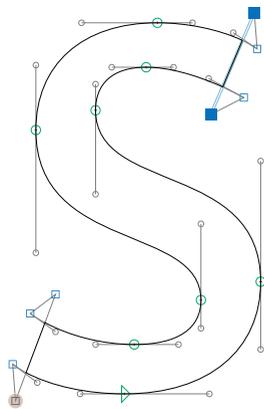


In order to recreate overlaps in a path, select an even number of nodes and choose *Reconnect Nodes* from the context menu. To open the context menu, right-click or Ctrl-click anywhere in the canvas. Glyphs will then proceed to reconnect each

node with its closest selected neighbor. The *Open Corner* command creates a triangular overlap at each selected point. Resegmenting your outlines this way allows you to manipulate the path segments independently from each other. It can also make interpolation significantly easier.

The size of the created overlap will be approximately half the first values entered for vertical and horizontal stems in *File > Font Info > Masters* (Cmd-I). Thus, the created overlap should comfortably reach into the middle of your stems.

Opened corners will be considered invisible if the triangular overlaps are small enough in relation to the neighboring visible outline segments. That way, opened corners can also be placed on the outside of paths. If the overlap size goes beyond the threshold size, then they will become visible again. This is useful for editing bent finials, as in a sans-serif lowercase *s*.

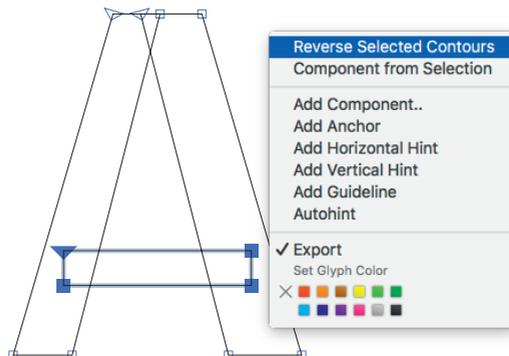


3.3.12 Controlling Path Direction

The starting point of a closed path, indicating the path direction. Again, green and blue denote a smooth connection or a corner, respectively. Likewise, the end points of an open path are displayed as arrowheads. On a closed path, you can make any on-curve node the first node by picking *Make Node First* from its context menu.

All outer (black) paths need to run counter-clockwise, while (white) counters must go clockwise. You can change a path's direction by selecting it and choosing *Paths > Reverse Contours* or *Reverse Selected Contours* from the context menu. One node on each path will suffice as path selection in this case. When no path is selected, you can use *Paths > Reverse Contours* to toggle all path directions in a glyph.

Tip: The easiest and quickest way to get a hole in a glyph is to draw both inner and outer shapes, and press Cmd-Shift-R (Correct Path Direction).



Paths > *Correct Path Direction* (Cmd-Shift-R) will perform an informed guess and find the right path orientation for all contours in selected glyphs. This will also rearrange the contour order, and reset the starting points of all paths to the bottom left nodes.

Holding down the Option key changes the command to *Correct Path Direction for all Masters* (Cmd-Opt-Shift-R). As the name indicates, it will include all master layers, but also all Bracket (see 12.8, 'Bracket Layers', p. 149) and Brace layers (see 12.7, 'Brace Layers', p. 149). The command ignores all other non-master layers. This is useful in a Multiple Master setup.

In order to successfully interpolate, path order, starting points, and path directions need to be compatible and consistent throughout all font masters. For more details on interpolation, see chapter 12, 'Multiple Masters' (p. 141).

3.3.13 Extremes and Inflections

Extrema are all positions on a path with a completely horizontal or vertical tangent. An inflection is the position in some path segments where the segment changes its bend from clockwise to counterclockwise or vice versa. It is considered good practice to have nodes on extremum points. Some font technologies, like hinting, and some path operations, like nudging (see 3.3.2, 'Moving Selected Nodes and Paths', p. 18), require nodes at extremum positions. Some operations, like offsetting a curve (see 5.2.3, 'Offset Curve', p. 58), work better with inflection points. Also, some font renderers may behave unexpectedly if such nodes are not in place. Inflection points pose a problem for outline interpolation, since they can easily cause kinks in outlines. Also, you may want to avoid additional nodes in order to keep the overall file size as small as possible, e.g., for webfont production.

You can insert nodes on extremum and inflection points by Shift-clicking a segment with the Path tool (P). A node will be inserted at the nearest extremum or inflection.

Alternatively, you can also choose *Paths* > *Add Extremes* and nodes will be added at extremes on all paths of the active layer. Glyphs will not add an extreme if the resulting segment would be very short. In this case, it assumes that the node placement was intentional. On the other hand, if a node is only very slightly off the extremum position, Glyphs will attempt to preserve the outline shape while moving the node

into the extremum position and turning the surrounding handles entirely vertical or horizontal.

You can have extremes added automatically at export time with a Filter custom parameter called 'AddExtremes'. See the Filter entry in the list of custom parameters in the Appendix (p.184) for details. This can be useful for shallow curves or certain Multiple Master situations, where adding extremes would make editing or interpolating unnecessarily complex.

3.3.14 Duplicate Nodes

When two adjacent on-curve nodes share the same coordinates, thus producing a zero-length segment, they are highlighted with a red circle. You can merge these nodes with *Paths > Tidy Up Paths* (Cmd-Opt-Shift-T).



3.4 ANCHORS

3.4.1 Compound and Positioning Anchors

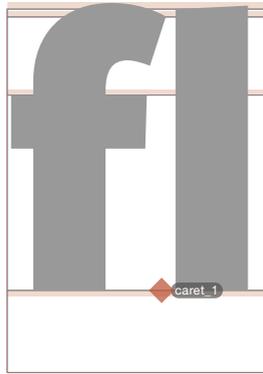
Anchors are special points that fulfill multiple tasks in Glyphs. Primarily, they serve as a connecting pivot for automatically aligning components, corners and caps, as well as for Mark-to-Base and Mark-to-Mark Positioning, and Curative Attachment (see 8, 'Reusing Shapes', p.98). These anchors adhere to certain naming conventions. For more details on these uses, see section 8.1.6, 'Anchors' (p.101). When an 'origin' anchor is placed inside a glyph, it can be used for aligning a component (see 3.3.6, 'Aligning', p.21) to a regular path node.

Some third-party scripts and plugins make use of special anchors. Refer to their documentation for further details.

3.4.2 Ligature Carets

Ligature caret positions are the positions where the text entry cursor is displayed when placed somewhere inside a ligature. In a ligature glyph, these positions are defined by special anchors on the baseline. They must be named 'caret', followed by an underscore suffix, e.g., 'caret_1', 'caret_2', etc. The suffix needs to be different for each anchor, because anchor names must be unique inside a glyph layer. The numbering

order does not matter, the numbers are used exclusively for differentiation.



Glyph > Set Anchors (Cmd-U) will insert appropriate caret anchors in properly named ligature glyphs, i.e., the names of the characters in the ligature, connected by underscores, e.g., s_t or f_f_h. For the glyph naming convention employed by Glyphs, see section 6.6, 'Names and Unicode' (p. 78). At export, Glyphs will use the caret information to build so-called *LigatureCaretByPos* instructions in the GDEF OpenType table. At the time of this writing, the only known software supporting ligature caret positioning are Mac applications that make use of the Cocoa text engine. Adobe and Microsoft apps ignore this information.

3.4.3 Adding, Editing and Removing Anchors

Insert an anchor by Ctrl-clicking or right-clicking inside a glyph, and choosing *Add Anchor* from the context menu. An anchor called 'new anchor' will be placed at the click position. Its name is immediately selected for renaming, so you can enter the name right away. Confirm the new name by pressing the Return key or clicking anywhere in the canvas.

To change an anchor name, select it, press the Return key, and type a new name. Alternatively, click or double click its name in the gray Info box when the anchor is selected, and type a new name there.

The built-in glyph info database has default anchors associated with some glyphs. To have them added automatically, choose *Glyph > Set Anchors* (Cmd-U) or, with

the Option key held down, *Glyph > Reset Anchors* (Cmd-Opt-U). The latter will delete all existing anchors before inserting the defaults.

Select an anchor by clicking on the red dot that represents it. You can select the next or previous anchor by pressing the Tab key, or Shift-Tab, respectively. Select multiple anchors by Shift-clicking them. Names are only shown for selected anchors. If you want to select all anchors, you may need to run *Edit > Select All* (Cmd-A) twice, since selecting all will select all paths, and only select all anchors and components if all available paths have been selected already.

Move an anchor like you would move a node, i.e., either with the mouse, the arrow keys, or through the gray Info box. An anchor is diamond-shaped if it is placed exactly on a metric line such as the x-height or the baseline. It is circle-shaped in all other cases. To quickly duplicate an anchor, Option-drag it. Since anchor names must be unique inside a layer, an underscore will be added to the end of the name.

Remove one or more selected anchors by simply pressing the Delete or Backspace key.

3.4.4 Mark to Base Positioning

Glyphs can automatically build the ‘mark’ (Mark to Base) feature from combining (non-spacing) diacritical marks containing underscore anchors, e.g., ‘_bottom’ or ‘_top’, in combination with all base glyphs that carry corresponding base anchors, e.g., ‘bottom’ or ‘top’. Latin combining diacritical marks carry a ‘comb’ at the end of their names, e.g., acutecomb or macroncomb.

Combining diacritical marks have their own Unicode values and thus can be typed or inserted in a text. This way, a user can place any mark on any base letter, by first typing the regular letter, and then inserting the combining mark.

Alongside the ‘mark’ feature, Glyphs will also build the ‘ccmp’ (Glyph Composition and Decomposition) if glyphs like idotless and jdotless are present. See chapter 17.1, ‘Automatic Feature Generation’ (p.175), for further details.

3.4.5 Mark to Mark Positioning

If both underscore and regular anchors are present in a combining diacritical mark, Glyphs will also automatically build the ‘mkmk’ (Mark to Mark) feature. A user will then be

able to stack any combining mark on any other combining mark carrying both anchors.

3.4.6 Cursive Attachment Anchors

To enable true cursive attachment in Arabic typesetting, add anchors called 'exit' and 'entry' to the respective stroke endings and beginnings in medial, final, and initial letter forms. The entry anchor of the instroke will be connected to the exit anchor of the preceding outstroke. You can preview cursive attachment immediately in the Edit view when right-to-left typesetting is enabled (see 3.8.3, 'Writing Direction', p. 38).

3.5 GUIDELINES

3.5.1 Magnetic Guidelines

If you drag a node selection across the canvas, red lines will appear, indicating when your selection is aligned with other nodes or a vertical metric. You can temporarily deactivate magnetic guidelines by holding down the Ctrl key.

Likewise, dragging a node or any other object will snap to all nodes and handles on paths, as well as in components. When you move close to a node in a component while dragging, Glyphs will fade in small representations of the nodes inside the component. Again, you can disable node snapping by holding down the Ctrl key.

3.5.2 Local and Global Guidelines

To toggle the display of guidelines in Edit view, choose *View > Show Guides* (Cmd-Shift-L)

To add a static, local guideline to the currently displayed glyph layer, right- or Ctrl-click to open the context menu and choose *Add Guideline*. A horizontal local guideline will be added at the click position. If two nodes are selected while adding the guideline, it will be laid through the nodes. You can select it by clicking on it, alter its position by clicking and dragging its knob. Double click the circle to turn it perpendicular to its original orientation. Click and drag a selected guideline anywhere else but on the knob to rotate it.

To create a global guideline, i.e., a guideline that appears in all glyphs throughout the font master, hold down the Option key while navigating the context menu, and choose *Add Global Guideline*. To toggle a guideline between local and global, select

Pro User Tip: In order to quickly create a guideline in measurement mode, temporarily activate the Measurement tool by simultaneously holding down Ctrl-Opt-Cmd, and press the G key while dragging a measurement line.

it, and from the context menu, choose *Make Global Guideline* or *Make Local Guideline*, respectively.

Quickly duplicate one or more local or global guidelines by selecting them and holding down the Option key while dragging them to a new position.

You can lock one or more guidelines by choosing *Lock Guidelines* from their context menu. A locked guideline cannot be selected and will display a lock symbol instead of its knob. To unlock a guideline again, Ctrl- or right-click the knob and choose *Unlock Guide* from the context menu.

From top to bottom:
unselected and selected
local guides, unselected
and selected global guides,
locked local guide, and
locked global guide.



Select a guideline by clicking anywhere on it: a filled knob indicates a selected guideline. Press the Tab key to quickly select the next, or Shift-Tab to select the previous guideline. When a guideline is selected, you can move it using your arrow keys (add Shift or Cmd for larger increments), or by dragging its knob with the mouse, just like you would move a regular node. Change the angle by dragging the guideline anywhere off the node. You can also enter values for its position and its angle in the gray Info box (Cmd-Shift-I).

By default, a guideline will be set relative to the left sidebearing. However, if you change the alignment of the guideline by clicking on the Alignment icon  in the Info box, the guideline will stay fixed relative to the  right sidebearing or  relative to both sidebearings. This can be useful for slanted guidelines, especially when they are global or when the right sidebearing is changed. With the measurement checkbox , you can turn it into a measurement guideline. For more details, see section 3.9, ‘Measuring’ (p.38).



3.5.3 Glyph-Specific Undo History

In Edit and Font view, the Undo mechanism works on a glyph level. That means that every glyph has its own undo history.

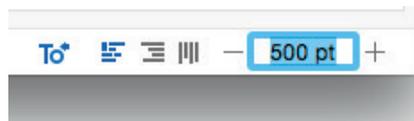
This also implies that certain global actions, especially manipulating global guidelines, cannot be undone. That is because global guidelines are associated with a master, and not a glyph, and therefore are ignored by the glyph-level undo history.

3.6 GLYPH DISPLAY

3.6.1 Zooming

There are many ways to zoom in and out in Edit view. If you are on a MacBook or have a trackpad to your disposal, you can use pinch and stretch gestures. Or hold down the Option key and use a scroll gesture or the scroll wheel of your mouse. Or activate the Zoom tool  (shortcut Z) and click in the canvas to zoom in, Opt-click to zoom out. Alternatively, click and drag across an area, and it will be zoomed to fill the window. You can temporarily activate the Zoom tool by holding down Cmd-Space for zooming in, or Opt-Cmd-Space for zooming out. If Cmd-Space collides with another shortcut on your system, you can try pressing Space before you add the Cmd key.

Or you can use the zoom commands from the *View* menu: *Zoom In* (Cmd-plus) and *Zoom Out* (Cmd-dash). *Zoom to Active Layer* (Cmd-zero) will maximize the area between ascender and descender in the window. *Zoom to Actual Size* (Cmd-Opt-zero) will zoom one font unit to the size of one screen point, i.e., one pixel on a classic low-resolution screen, two pixels on the new high-resolution Retina screens.



Or you can use the zoom buttons in the bottom right corner of the window. Alternatively, you can set the zoom value numerically by entering a point height in the field between the buttons. The value specifies at which size 1000 font units will be displayed. Since Mac OS X assumes a screen resolution of 72 ppi, one point corresponds to one actual screen pixel, so if you enter a value of 1000, one unit will zoom to one pixel. On Retina displays, zooming is done according to their

Tip: You may need to change your Spotlight shortcut in the System Preferences for the Cmd-Space shortcut to work in Glyphs.

higher resolution, e.g., at 144 ppi, a unit will be zoomed to two screen pixels.

3.6.2 Panning

You can use trackpad panning gestures, i.e., dragging two fingers across your trackpad, to change the canvas excerpt of the window. Use the wheel of your mouse to scroll vertically, hold down Shift to scroll horizontally. Or, drag the scroll bars on the right and bottom edges of the displayed canvas. If you have difficulties grabbing the scroll bars, consider changing the scroll bar behavior in the *General* section of System Preferences. Alternatively, you can switch to the Hand tool  (shortcut H) and drag the canvas around, or simply hold down the space bar to temporarily switch to the Hand tool. If you are in text mode, simply pressing the space bar would add a space to your sample text. To avoid that, you can press Cmd-Space and subsequently release the Command key.

Tip: Cmd-Space may interfere with the system-wide shortcut for invoking Spotlight. You can change it in the System Preferences.

3.6.3 View Options

In the *View* menu, you can toggle several options that influence the glyph display in Edit view.

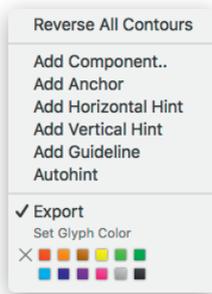
- *Show Nodes* displays on- and off-curve nodes of paths,
- *Show Metrics* shows the vertical and horizontal metrics of the glyph,
- *Show Hints* displays PostScript hints,
- *Fill Preview* displays closed paths with a black fill.

Many third-party reporter plugins are available for changing or enhancing the glyph display in Edit view. After installation, they will also show up in the *View* menu. See section 16.2, ‘Plugins’ (p. 172), for more details.

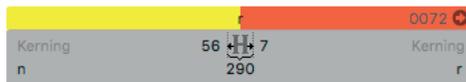
3.6.4 Glyph and Layer Colors

In Edit view, you can set both glyph-wide and layer-specific label colors via the context menu. Right- or Ctrl-click

anywhere in the canvas of an active glyph, and pick the color from the context menu.



If you hold down the Option key, the *Set Layer Color* option becomes active, and picking one of the colors will change the layer color rather than the glyph color. Both glyph and layer colors will be displayed in the gray Info box (*View > Show Info*, Cmd Shift-I):



The glyph color is displayed on the left half, the layer color on the right half. This corresponds with the display of label colors in Font view. See section 6.3.5, 'Color Label' (p. 71).

3.7 BACKGROUND

Each layer has a background layer, usually simply referred to as 'background'. The background is useful for temporarily storing a path, or for tracking changes and comparing outlines before and after a manipulation. Some filters, such as *Filter > Hatch Outline*, use the background as backup layer in order to work non-destructively.

While working in the foreground, objects on the background can be displayed as subtle gray outline with *View > Show Background* (Cmd-Shift-B). If this option is active while you are working in the background, the foreground objects will be displayed this way. When the background is displayed, snapping will also work with objects on the background layer.

To switch to the background, choose *Paths > Edit Background* (Cmd-B). The window display will darken slightly to indicate

that you are in the background layer. You can set a different color in *Glyphs > Preferences > User Settings*.

Paths > Selection to Background (Cmd-J) replaces the current content of the background with the active selection; this works in reverse when the background is active. Simultaneously holding down the Option key changes the command to *Add Selection to Background* (Opt-Cmd-J), and copies the current selection to the background without clearing it first. *Paths > Swap with Background* (Ctrl-Cmd-J) will exchange the foreground with the background. You can empty the background layers of selected glyphs by holding down the Option key and choosing *Paths > Clear Background*. When applied to multiple glyphs at once, these commands work on complete layers, i.e., ignoring individual path selections.

Via *Paths > Assign Background*, you can copy the outlines of another font file into the background layer of all selected glyphs. You can also put the same font into its own background in order to keep track of your own changes. Selecting all glyphs and choosing *Paths > Selection to Background* (Cmd-J) has the same effect.

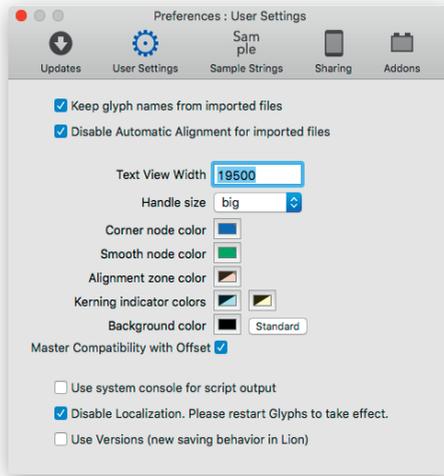
3.8 ENTERING TEXT

The Edit view also acts as a simple text editor, and as such, allows you to edit your glyphs in the context of a whole word or even a sentence. As long as the Text tool (T) is active, it accepts input via the current input source set in the System Preferences, the Keyboard Viewer, the Character Viewer, but also via the clipboard or *Edit > Special Characters*. With *Edit > Start Dictation* (Cmd-Cmd), you can even use the speech recognition features provided by OSX and dictate your text.

The Edit view has a preset linewidth. You can set the maximum linewidth in the application preferences. Select *Glyphs > Preferences > User Settings* and edit the *Text View Width* value accordingly. The value is in design units. Their relation

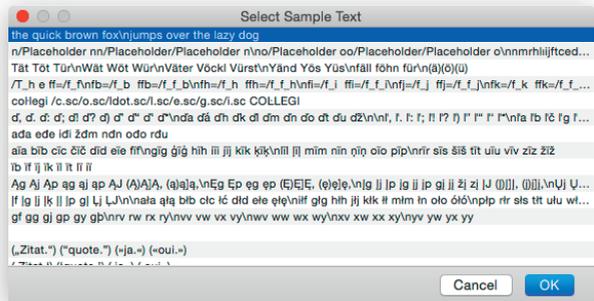
Tip: To quickly switch back from text entry to editing the current glyph, press the Esc key.

to the em size is defined by the *Units per Em* value in *File > Font Info > Font*.



Keep in mind that a lot of text in the Edit area will slow down the application, especially if the Preview is displayed. If you want to test your font with large chunks of text, it is advisable to use the Adobe Fonts folder, see section 3.12.7, 'Previewing in Adobe Applications' (p.49) for further details.

3.8.1 Sample Texts



You can edit and store any number of sample texts in *Glyphs > Preferences > Sample Strings* (see 2.1.3, 'Sample Strings', p.13). To make an Edit tab display any of these texts, choose *Edit > Select Sample Text* (Opt-Command-F). In the appearing dialog, use

your arrow keys or click on a line to pick the sample text and press Return or click the OK button afterwards.

To switch to the next or previous sample string without invoking this dialog, choose *Edit > Other > Select Next Sample String* or *Select Previous Sample String*, respectively. You can assign keyboard shortcuts in System Preferences.

3.8.2 Text Tool

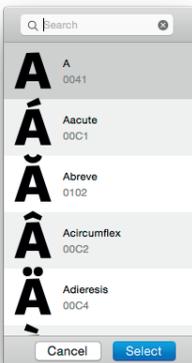
Select the Text tool (shortcut T) to switch to text mode and start typing. You can enter multiple words, complete sentences, even line breaks. You can copy and paste text into and from the Edit tab. The arrow keys, the *Edit* commands and OS X Application Services work as they would in any Mac app.

The current glyph is the one to the right of the cursor. You can switch to the previous or next glyph in the font by pressing the Home and End key, respectively. Add Shift to advance through the glyphs as they are currently visible in the Font tab. This is useful when you filter glyphs in the Font view and then want to step through each one of them.

Edit > Add Placeholder (Cmd-Opt-Shift-P) inserts a placeholder for the current glyph. Placeholders are dynamically replaced by the currently selected glyph. This is useful when editing sidebearings, and you want to see the same glyph between others and next to itself at the same time. E.g., you can switch from ‘ononnoon’ to ‘omommmoom’ if the n glyphs are placeholders.

To insert one or more glyphs that you cannot or do not know how to type with the keyboard, choose *Edit > Find > Find...* (Cmd-F). In the dialog that appears, enter the glyph name, a part of it, or different parts of the glyph names separated by spaces. The dialog will show a list of available glyphs whose name contain the strings you entered. E.g., ‘dier o1’ will find all glyphs that have both ‘dier’ and ‘o1’ in their name, like *adieresis.sso1* and *edieresis.cvo1*. Select the glyphs you want and press Return or click the Select button to insert it into your sample text. To select a range of glyphs, hold down the Shift key. For a non-contiguous selection, hold down the Cmd key.

Tip: On a MacBook, you can simulate Home and End by pressing Fn-leftarrow and Fn-rightarrow.



3.8.3 Writing Direction

In Edit view, switch between left-to-right, right-to-left, and top-down with the respective alignment buttons in the bottom right corner of the application window.

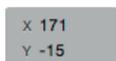


3.9 MEASURING

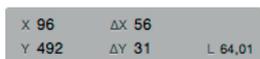
Glyphs offers several ways to determine coordinates and to measure distances between points and curves.

3.9.1 Info box

You can toggle the display of the gray Info box with *View > Show Info* (Cmd-Shift-I). The Info box always displays data relevant to the current selection. If there is exactly one node selected, its coordinates will be displayed.



Select a handle (off-curve point, Bézier control point), and the Info box will also show its delta values (x and y difference to the on-curve point) and the total length of the handle (distance to the on-curve point).



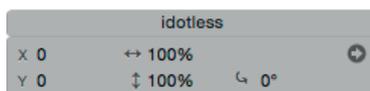
Tip: To quickly and precisely measure a stem or bowl width, select two nodes that indicate the width and see what the Info box displays next to the width symbol.

If you select more than one point, be it on- or off-curve points, the Info box will display the width (↔) and the height (⇅) of the box defined by the current selection. The x and y coordinates displayed describe the position of the bounding box of the selection as indicated by the grid, i.e., the bottom center point of the grid indicates that the coordinates describe the bottom center point of the bounding box. The two numbers to the right indicate the count of nodes in the current glyph, and in the current selection, respectively.



In the Info box, any displayed number (except for the node counts) can be selected and changed by entering a new value. You can use the Tab key to advance to the next number slot displayed, and Shift-Tab to switch to the previous slot. Changes take effect once you press Return or tab out of a value field. Alternatively, you can use the up and down arrows for increasing and decreasing the number by one unit. Add Shift for increments of 10. If you scale a selection this way, the lock symbol toggles proportional x/y scaling, and the grid controls the scaling origin.

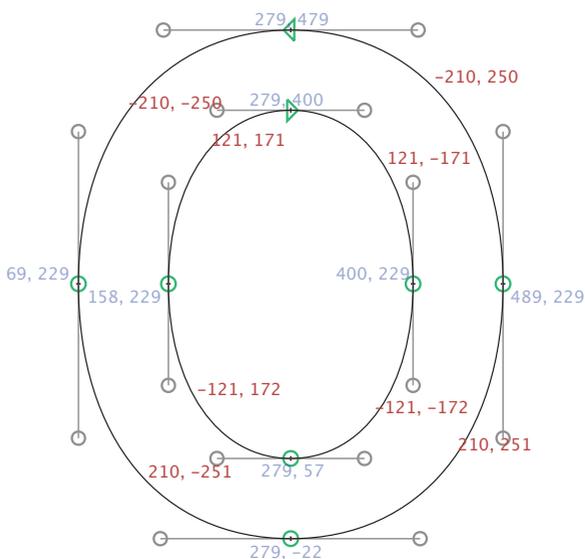
When a component is selected, the Info box will display the name of the original glyph the component points to, its x and y offset, its horizontal and vertical scale in percent, and its counter-clockwise rotation angle. The little arrow button in the top right corner will insert the original glyph in the Edit tab string, to the left of the current glyph, and activate it for editing. For more details on working with components, see section 8, 'Reusing Shapes' (p. 98).



You can change the glyph the component points to by clicking on its name in the gray Info box, and choosing another glyph from the glyph list in the subsequent pop-up window. Change position, size, and rotation by manipulating the respective number values. Use the up and down arrows to step through values, and add Shift or Cmd for increments of 10 or 100, respectively. Changing the position only has an effect if the component is not automatically aligned. For more details on automatic alignment, see section 8.1.7, 'Automatic Alignment' (p. 103).

3.9.2 Measurement Tool

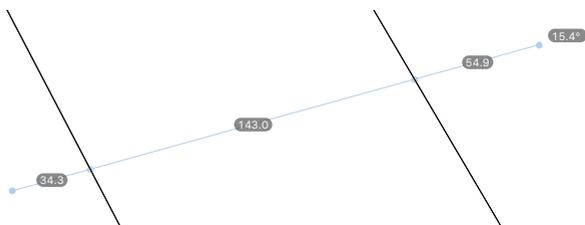
Switching to the Measurement Tool  (shortcut L) allows you to see all coordinates of all nodes and anchors at once.



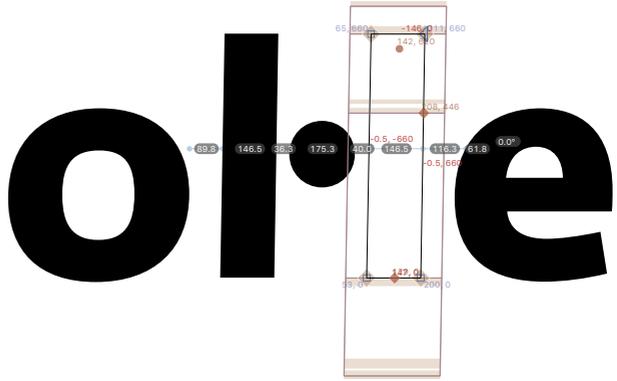
The x delta values respect the italic angle set in Font Info. So, an x delta of zero indicates a line exactly in the italic angle.

The blue numbers are the x and y coordinates of the on-curve points, the red numbers are the x and y delta values between the on-curve points. These values help you check the symmetry of your curves.

Clicking and dragging draws a ruler that displays precise measures between all of its intersections with outlines. Add Shift to drag a horizontal or vertical ruler. At the end of the ruler, its angle is displayed in counter-clockwise degrees, where zero degrees corresponds to dragging the ruler completely horizontally to the right.



The Measurement Tool works on all visible glyphs in Edit view. So, you can drag a measurement line across multiple glyphs and see the individual distances:



You can temporarily activate the ruler and the display of point coordinates by simultaneously holding down Ctrl, Option and Command. Pressing the G key while dragging a ruler adds a guideline in measurement mode.

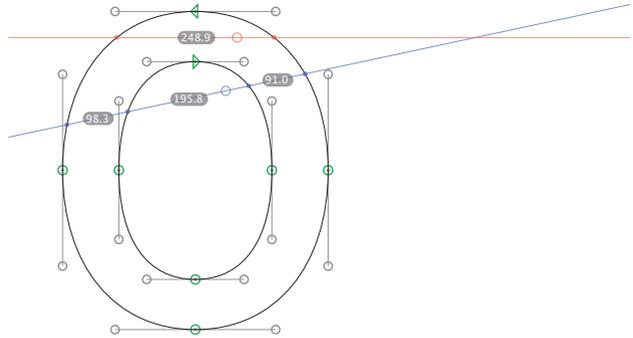
3.9.3 Measurement Guidelines

Any guideline, even a global guideline, can be turned into a measurement guideline. Simply click on the guideline to select it, and then, click on the measurement symbol  in the gray Info box. You can handle measurement guidelines exactly as regular guidelines, see section 3.5, 'Guidelines' (p.30) for more details.



Similar to the Measurement tool, guidelines in measurement mode will display the distance between their intersections with outlines or components. Contrary to the tool, they

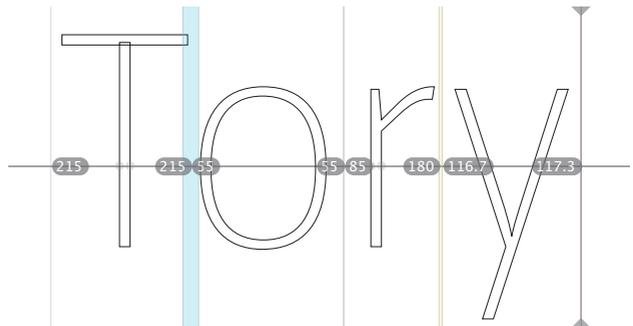
always do so as long as guidelines are shown, no matter which tool is active.



3.9.4 Measurement Line

When you are in text mode, you can enter measurement mode by choosing *View > Show Measurement Line*. The measurement line will display the sidebearings at a given height, ignoring the shape of the glyph at other positions. More precisely, the numbers displayed indicate the distance between the left or right sidebearing and the point where the measurement line first crosses the glyph outline. You can alter its height by Ctrl-Opt-Cmd-clicking or Ctrl-Opt-Cmd-dragging. Or you can switch to the Measurement tool (L) and simply drag it to the desired height.

In measurement mode, thin gray lines indicate the widths of the glyphs. Kernings receive a color code. Negative kernings will be displayed light blue, positive kernings yellow. You can change the colors in *Glyphs > Preferences > User Settings*.



3.10 ANNOTATING

The Annotation tool  (shortcut A) allows you to add simple notes and correction marks to your drawings. Once you activate the tool, the gray Info box (*View > Show Info*, Cmd-Shift-I) will turn into a little palette holding a range of annotation tools.

Choose *Edit > Select All* (Cmd-A) to select all annotations in the currently active glyph layer. Move selected annotations with the cursor keys. Hold down Shift for increments of 10, and Cmd for increments of 100 units. Press the Delete key to remove all selected annotations.

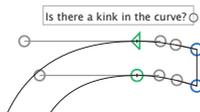


3.10.1 Annotation Cursor

The first tool on the Annotation palette is a simple edit tool for annotation marks. It allows you to activate, move, and resize existing annotations. Shift-click to select more annotations. Once an annotation mark is activated, you can delete it by pressing the Delete key.

3.10.2 Annotation Text

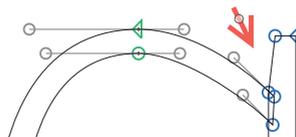
The second tool from the left is a simple text tool. Activate it by clicking on the T button and then click anywhere on the canvas to add a text box. Double click on the text and start typing. When you're finished, just activate the next tool you want to work with. You do not need to acknowledge the text entry.



The handle on the right controls the width of the text box. The height of the box always automatically adjusts to the length of the entered text.

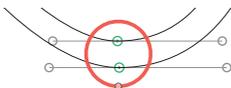
3.10.3 Annotation Arrow

The third tool on the Annotation palette allows you to put red arrows on the canvas. The handle on the arrow stem controls the rotation of the arrow.



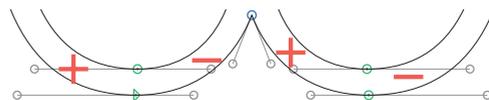
3.10.4 Annotation Circle

The fourth tool puts red highlighting circles on the canvas. The handle at the bottom of the circle controls the diameter.



3.10.5 Plus and Minus Annotations

Many designers use plus and minus signs to indicate that a counter, a bowl or a stem needs to be thickened or thinned, respectively. Click on the plus or minus button and then on the canvas to add the symbols to the editing area.



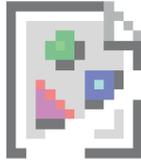
3.11 IMAGES

3.11.1 Adding Images

You can add all image files supported by OS X, including PDFs, to any glyph layer simply by dragging them onto a glyph cell in Font View, or into the active glyph in Edit View. You can add many images at once by choosing *Glyph > Add Image* and selecting any number of image files in the subsequent Open File dialog. The images will be placed in the current master layers of those glyphs that correspond to the file names. E.g., if your scans are called *Thorn.png* and *thorn.jpeg*, they will be placed in the glyphs *Thorn* and *thorn*, respectively. For this to work, you may have to place scans for uppercase and lowercase letters in separate folders. That is because, by

default, the OSX file system is set to be case-insensitive, and thus, same names with different capitalizations cannot coexist in the same OSX folder.

In the Glyphs document, only the relative path of the image is stored. Thus it is a good idea to keep them in a subfolder next to the Glyphs file. If the path to the placed image is outdated or broken, it will be indicated with a missing image symbol:



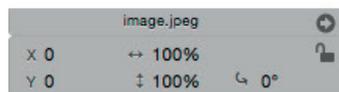
You can toggle the display of images via *View > Show Image*. Even if this setting is off, images will be displayed as long as the glyph is empty, i.e., contains neither paths nor components.

Image files will be ignored at OTF export unless you export a bitmap image font, like an Apple-style color font. See section 13.4, 'Apple Color Fonts' (p.155) for details.

3.11.2 Manipulating Images

By default, images will be scaled to a size where one DTP point corresponds to one font unit, and placed at the origin point of the layer. If you plan to place a lot of images, consider preparing their size accordingly.

Move the image by dragging it to the desired position. When an image is selected, you can resize it with the bounding box (Cmd-Opt-Shift-B) or the Scale tool (S). You can rotate it with the Rotate tool (R). The *Transformations* palette also works for images.



The gray Info box (Cmd-Shift-I) allows you to numerically control position (x and y) and dimensions (↔ for width, ↕ for height) of the selected image. You can also rotate the image counter-clockwise by manipulating the degree figure next to

the curved arrow symbol. A click on the right-pointing arrow symbol will reveal the original image file in Finder. A click on the lock symbol freezes the image status until you unlock it again via the context menu.

Besides locking and unlocking an image, the context menu lets you crop the image to the layer bounds (width, descender, and ascender), and reveal the image file in Finder.

3.12 PREVIEWING AND TESTING

3.12.1 Previewing Kerning

Kerning Preview is on by default. You can activate and deactivate kerning with the Preview Kerning button in the bottom right corner of the window. Clicking the button toggles between **T o** no kerning, **T o^{*}** kerning, and **T o^l** locked kerning. The no-kerning setting disables the display of kerning altogether. The kerning and locked-kerning settings both show kerning, but the latter disables the editing of the spacing. This can be useful if you want to edit the kerning without accidentally manipulating spacing.

3.12.2 Previewing Masters

The Edit view already is a preview of the master(s). It always previews the anti-aliased outlines and the kerning of the currently selected font master. When the Glyphs file contains more than one font master, a button row representing each master is displayed in the top left of the window. Switch between masters by clicking on these master buttons, or by pressing **Cmd** and the number of the master you want Glyphs to display, e.g., **Cmd-1** and **Cmd-2** for the first and second master, respectively.



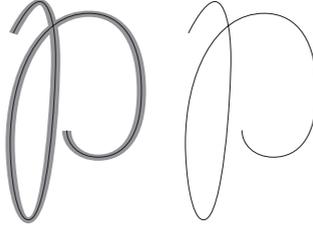
For more details on working with multiple font masters, see chapter 12, 'Multiple Masters' (p. 141).

3.12.3 Previewing Path Offset

Via **View > Show Preview Offset**, you can toggle a gray preview of the offset the *Offset Curve* filter would produce with its current settings. This is useful for an open paths workflow.

For the preview to take effect, *Filter > Offset Curve* must have been run at least once. The preview only applies to the current glyph.

A lowercase script p with Preview Offset enabled (left) and disabled (right). The offset value is taken from the settings used the last time the Offset Curve filter has been run.



3.12.4 Previewing OpenType Features

You can activate and deactivate OpenType features through the *Features* menu at the bottom left of the window in Edit view. When at least one feature is selected, the *Features* menu will be highlighted. The button will show the four-letter tags of the active features.

You may need to recompile the features in *File > Font Info > Features* before they are available in the pop-up list. You can select any number of features concurrently, and deselect all of them at once by choosing the dash at the top of the menu.

To preview language-specific forms, first choose *Localized Forms* from the menu. Subsequently, you can preview a script or language at the bottom of the menu. For this to work, a 'locl' feature with valid language-specific rules must be present in *File > Font Info > Features*.

In Edit view, Glyphs will show a preview only of substitution features, since positioning features can be handled very differently in different application, application settings, and system environments. To test positioning features other than kerning, it is therefore recommended to make use of the Adobe Fonts folder. For details, please see section 3.12.7, 'Previewing in Adobe Applications' (p. 49). If you preview in InDesign, note that OpenType features may be interpreted differently in different compositors.

3.12.5 Previewing Interpolated Instances

Click on the Preview button (with an eye symbol), next to the Features drop-down menu. The window content will be sectioned vertically, with the Edit view at the top, and the

Preview Area at the bottom. Drag the separator line to adjust the size of the Preview.

Alternatively, you can open a separate window via *Window > Preview Panel*. You can move the Preview Panel to a separate display connected to your Mac. The controls will fade out once you move your mouse out of the window. The Preview Panel will display a preview for the current Edit tab of the current font file. Since the panel is detached from the document windows, it can go blank if no active Edit tab can be determined. In that case, simply click into an Edit area to activate it, and the Preview Panel will update its contents.

To see a live interpolation of an instance, pick an instance from the instances pop-up below the Preview Area, or at the bottom of the Preview Panel. The menu item *Show All Instances* renders the current glyph in all active instances next to each other. You can activate and deactivate instances in the Font Info, see section 7.3, 'Instances' (p. 90) for details. Once you select the dash at the top of the menu, the Preview will be reset to a rendering of the current master rather than an instance.

With the exception of the *Show All Instances* option, the Preview Area and Preview Panel render the complete text of the current Edit tab on a single line, and center on the current glyph. You can double click a previewed letter to make the Edit view center on it. The rendering respects some custom parameters, as well as the brace and bracket tricks described in the section 4.4.2, 'Special Layers' (p. 53).

Right-clicking into the Preview area yields a context menu that gives you the option *Always Center Active Glyph*. When activated, the currently active glyph appears centered at all times. If deactivated, Glyphs will try to fill the Preview area as well as possible, keeping the text flush left or right.

You can switch between black-on-white and white-on-black with the Invert button  next to the instances pop-up. With the 'F' button, you can flip the Preview upside-down. This can be helpful when testing the spacing of your font. To test the legibility of your design, you can use the slider to blur the font sample in the Preview.

3.12.6 Previewing in OS X

Because of the complicated font caching mechanism employed by OS X, simply overwriting a previously installed font can

Because of the described difficulties with the OS X font cache, we do not recommend testing unfinished fonts in Font Book.

lead to a range of problems. Amongst other things, the font may not show the changes you made in Glyphs, or may show wrong glyphs for the characters typed, display empty or garbled glyphs, or even disappear from the font menu. In order to avoid such font cache difficulties, you can change the family name in *File > Font Info > Font* (Cmd-I) every time you export, e.g., by adding an incremental number ('MyFont 01', 'MyFont 02', 'MyFont 03' etc.) or a letter combination ('MyFont AA', 'MyFont AB' etc.). This scheme ensures a more consistent font menu ordering if you have many versions of your font installed.

If you do run into cache problems, close all running applications and remove all instances of your font from Font Book or the Fonts folder. Then, open the Terminal application and type these lines, each of them completed by pressing the Return key. The first command will prompt you for your administrator password. You will not see bullets while you type your password.

```
sudo atsutil databases -remove
atsutil server -shutdown
atsutil server -ping
```

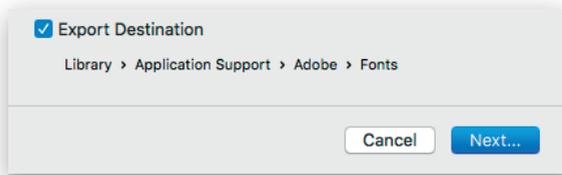
After these Terminal commands, you need to restart the Mac for the changes to take effect. If the problems persist, boot the Mac in safe mode, i.e., hold down Cmd-S at startup until the Apple logo appears on screen. Recreate caches by holding down the Shift key while restarting and logging in.

3.12.7 Previewing in Adobe Applications

For a complete preview including things like positioning features and menu ordering, just pick *File > Export* (Cmd-E), pick the *OTF* export, and choose */Library/Application Support/Adobe/Fonts/* as *Export Destination*. The font becomes immediately available in all Adobe applications. Any previously saved instance of the font in this folder will be overwritten. The font will not be available outside Adobe apps, but this is a convenient way to circumvent any font cache problems in OS X.

If the Fonts folder does not exist, you can create it right in the *Open Folder* dialog by pressing Cmd-Shift-N. Only in case they were already running while the folder was created, you have to restart Adobe apps this once for the change to

take effect. After that, the Adobe font menus will update immediately every time the font is exported.



3.12.8 Previewing in Web Browsers

You can also export webfonts via *File > Export > Webfonts*. For repeated exports, you can set an *Export Destination* in the same dialog. If you set up an HTML file containing a test page that uses the webfonts. Open the file in a web browser, and reload the page every time a new version of the font was exported. Since the webfont files are overwritten during the export, reloading the page in the web browser should show the new version of the font. Some web browsers also employ font caches, though. If a reload does not show the new font, most Mac browsers offer you to force a clean reload of the page by holding down the Shift key while clicking on the reload button, or in addition to the keyboard shortcut for reloading (usually Cmd-R).

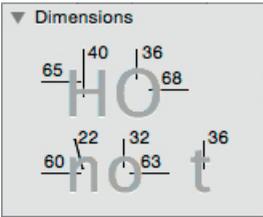
4 Palette

4.1 PALETTE SIDEBAR

You can open the Palette sidebar with the sidebar button in the top right corner of the window, or via *Window > Palette* (Opt-Command-P). By default, the Palette has four sections. You can collapse or expand them by clicking on their title or the triangle next to it. The sidebar can be extended with third party palette plugins.

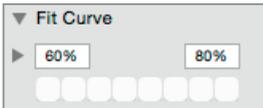
4.2 DIMENSIONS

The entries in the *Dimensions* section have no effect on the font. They serve as a cheat sheet for your personal design process. You can enter values for a couple of crucial measures in your current layer. Values are stored per master. For non-Latin scripts, such as Thai or Devanagari, appropriate dimensions are displayed as soon as a glyph of the respective script is selected in Font view or active in Edit view.



4.3 FIT CURVE

The *Fit Curve* panel helps creating curves with matching or corresponding curvatures. In most instances, the collapsed (i.e., single line) view will do. Select percentages by setting the left field to the minimum value and the right field to the maximum value. The eight buttons in between represent equal steps between these two values. Alternatively, you can use Ctrl-Opt-1 through 8 as keyboard shortcuts for the buttons. For the *Fit Curve* function to take effect, at least one handle must be selected. *Fit Curve* also works on multiple segments, provided that at least one handle of each segment is selected.



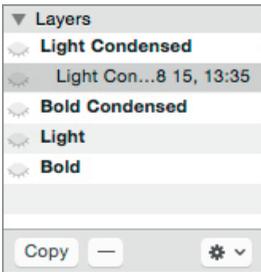
Smaller curves, e.g. counters, generally need higher curvature percentages than larger curves, e.g., on the outside of a bowl.

The numbers describe the percentage length of the handle. The distance from the curve point to the intersection of the handles equals 100 percent. 56 percent give you an elliptic curvature. The minimum value you can set is 1 percent, the maximum value is 100 percent. If you want a completely flat segment, select one of the handles and press the Delete key in order to turn the segment into a line.

For (rare) cases where you need finer tuning, click the little triangle to expand the *Fit Curve* panel into two dimensions. In that case, the first handle on each curve segment is controlled by the x axis, while the second one is controlled by the y axis.

The order of handles is controlled by the path direction of the path in question.

4.4 LAYERS



Glyphs differentiates between two sorts of layers: *master layers* and *normal glyph layers*. Master layers are needed for interpolating instances, or for keeping font variations. All glyphs in a font will always have layers for all font masters set in *File > Font Info > Masters*. In the *Layers* section of the Palette, master layers are displayed with the respective font master name in bold type.

Layers have an eye symbol next to them. It serves as a toggle button for the display of the layers. In its active state (indicated with an opened eye), the paths and components on the respective layer will be displayed with a gray outline behind the current layer, similar to background paths. Likewise, a closed eye indicates that the layer will not be displayed in the background.

Normal glyph layers are displayed in regular type, indented below the respective master layer they are associated with. You can have any number of normal glyph layers in a glyph. Use them for keeping variations of the respective glyph, for special techniques such as Brace or Bracket layers, or for creating color fonts. See chapter 13, 'Color Fonts' (p.151) for more details.

4.4.1 Working with Layers

Font master names cannot be edited through the Palette. They are controlled via *File > Font Info > Masters*.

Select a master layer and click the *Copy* button to duplicate it as a normal glyph layer. A copy of the master layer will appear below the master layer, carrying the name of the master and the creation date. Double click the layer name to edit it. To delete a layer, select it and click the minus button below. To swap it with the master layer, click the gear button and select *Use as Master* from the menu that pops up.

The *Re-Interpolate* menu item resets a selected layer on the basis of at least two other available master layers in the glyph. This is especially useful for Brace layers. See section 12.7, 'Brace Layers' (p.149) for details.

4.4.2 Special Layers

The name of a layer can determine a special function. Some layers have a special meaning in Multiple Master setups, namely so-called Brace layers (section 12.7, p. 149) and Bracket layers (section 12.8, p. 149). Chromatic fonts also require a special layer setup. See chapter 13, ‘Color Fonts’ (p. 151), for more details.

Third-party plugins may require special layers for their function. Refer to their documentation for further information.

4.5 TRANSFORMATIONS

The bottom section of the palette is reserved for object transformations. Once you have set the transformation origin, you can, from top to bottom:

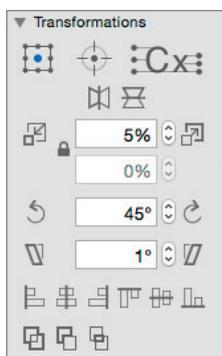
- mirror the selection  horizontally or  vertically,
- scale the selection  down or  up, horizontally and/or vertically, by percentage values,
- rotate the selection  counterclockwise or  clockwise by degrees,
- skew the selection  left or  right by degrees,
- align nodes, complete paths or components to each other,
- and execute boolean operations with two paths.

The transformations work in both the Font and the Edit view. Depending on the selection, the transformations work on paths, parts of paths, or complete glyph layers, even when multiple glyphs are selected. Note that, if you execute a transformation while a unit grid is active (see 7.5.1, ‘Grid Spacing and Subdivision’, p. 96), the placement of the on-curve nodes will be subject to rounding.

4.5.1 Transformation Origin

Except for aligning, all transformations available in the palette respect the transformation origin you set in the top row. This can either be a grid point calculated relative to the bounding box of the selection, similar to the grid in the gray Info box (Cmd-Shift-I).

Or, it can be  a manually set reference point. To change the reference point, you need to switch to either the Rotate (shortcut R) or the Scale (shortcut S) tool, and click once in the canvas.



Or, it can be  one of the metrics taken from the entries in *File > Font Info > Masters* (Cmd-I): baseline, (half or full) x-height and (half or full) cap height.

Tip: If you copy and paste the paths of the lowercase n into your lowercase u, you can mirror horizontally and vertically around half the x-height, rather than around the selection center. This way, the overshoot is respected.

If you want to scale down all coordinates by half, you need to enter 100 percent and click the  reverse-scale button, or enter 50 percent and click the  upscale button.

4.5.2 Mirroring

Mirroring can be applied to point selections, complete selected paths, and selected components, or any combination thereof. Mirroring a corner component will turn a left corner into a right corner and vice versa.

4.5.3 Reversible Transformations

The corresponding buttons to the left and right of the Scale, Rotate, and Skew fields, are each other's reverse transformations. Thus you can reverse any of the transforms with their complimentary buttons, and the points will assume their previous position, even when a unit grid is in effect and rounding has been applied to point coordinates.

Scaling takes two percentage values, for horizontal (x) and vertical (y) scaling, respectively. When the lock symbol is closed, the second value will be ignored and the first value applies to both x and y scaling. The entered values refer to positive scaling which is activated by pushing the  upscale button. This means that the  reverse-scale button does not scale down by the entered values, but reverses the positive transformation. This way, you can always undo a positive scale with the reverse-scale button and vice versa.

The Rotate and Slant functions work on any selection. Again, the buttons compliment each other, by reversing each others' transformations.

4.5.4 Aligning

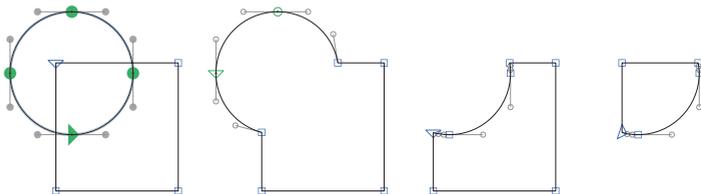
The Align buttons work on complete and partial paths, as well as components. If a path is selected only partially, the selected nodes are aligned to each other. Aligning via the buttons is always done relative to the bounding box of the selection.

For most use cases, a quicker way to align selected points is *Paths > Align Selection* (Cmd-Shift-A). This command respects the grid setting for the transformation origin. See section 3.3, 'Editing Paths' (p. 18), for more details.

4.5.5 Boolean Operations

The bottom row of buttons allows for boolean operations:  union,  subtraction, and  intersection. Union and intersection work with the selected paths, or all paths if none are selected. The subtraction operation subtracts the selected paths from the unselected ones. Or, if no paths are selected, it subtracts the front-most, i.e., last, path from all others on the glyph layer. In order to achieve consistent results, both groups of paths involved, i.e., either the selected and the unselected paths, or the front-most and all other paths, are merged before the subtraction is executed between them.

From left to right:
(1) two overlapping paths
before the boolean operation;
(2) union;
(3) subtract;
(4) intersection.



5 Filters

5.1 FILTERS

Filters process glyph layers. Their functionality ranges from simple width transformations to randomized distortions of the outlines.

5.1.1 Filters Menu

In Edit view, you can apply filters to the active layer, or, with the Text tool (shortcut T), to the displayed layers of any number of selected glyphs. In the Font view, you can apply it to all selected glyphs at once. Filters usually only affect visible glyph layers. So if you want to apply the same filter to all glyph layers, you will need to run the filter again for each master.

5.1.2 Filters as Custom Parameters

Most filters can also be applied to a font instance at export time by means of a custom parameter. To set up a custom parameter, select the instance in question in *File > Font Info > Instances*, add a parameter in the *Custom Parameters* section, switch its *Property* to 'Filter'. Then, set the *Value* to the parameter name of the filter in question, followed by semicolon-separated arguments: *parameterName; value1; value2*. Sometimes, the arguments are prefixed by an argument name, especially in filters where the arguments are optional: *parameterName; argument1: value1; argument2: value2*. You can limit the effect of the filter to certain glyphs with an additional *include:* argument followed by comma-separated glyph names. Analogously, you can apply the filter to all but certain glyphs with the *exclude:* argument, e.g., *GlyphsFilterOffsetCurve; 5; 5; 1; 0.5; exclude:a,b,c*. For better legibility, you can add whitespace characters around the arguments. You can add multiple filter entries to the custom parameters of an instance, and drag them into a specific order to combine their effects.

Most Filter dialogs sport a gear menu in their lower left corner of the window. You can quickly copy a custom parameter with the current values into the clipboard by choosing *Copy Custom Parameter* from the gear menu. Before pasting the parameter in *File > Font Info > Instances* (Cmd-I),

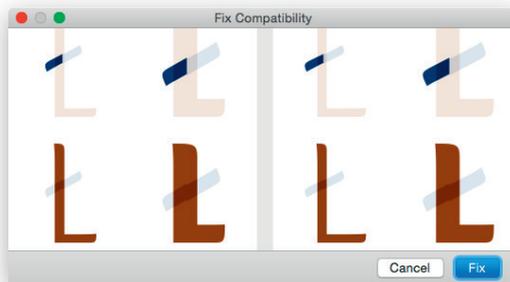
you need to click once in the *Custom Parameters* field to set the focus, i.e., the target for the clipboard operation.

5.2 BUILT-IN FILTERS

5.2.1 Fix Compatibility

This filter provides a graphical user interface for reordering paths and components. Paths are shown in blue, components in a rust color. Every line in the grid represents an object position, every column a master, bracket or brace layer of the glyph. Drag the objects vertically into the same order for all layers. When you are done, press *Fix* to confirm the new ordering of objects in the glyph.

If there are multiple, separated interpolations, they are separated by a vertical gray gutter. This can be the case with Brace layers (section 12.7, p. 149). It can also occur in more complex master setups, where, e.g., all condensed masters interpolate within each other, and all extended masters interpolate within each other, constituting two separate, unrelated lines of interpolation within the same glyph. *Fix Compatibility* is useful for cases where the automatic reordering through *Paths > Correct Path Direction for All Masters* (Cmd-Opt-Shift-R) does not yield the desired master compatibility. It is also useful when the display of *View > Show Master Compatibility* (Cmd-Opt-Ctrl-N) becomes too complicated, and it is too hard to keep the oversight. This can be the case in complex glyphs with many paths or components.



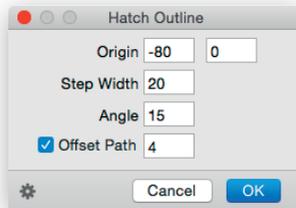
5.2.2 Hatch Outline

Creates hatched letters. Distance, width, and angle of the hatch-lines can be chosen. Glyphs always uses the background

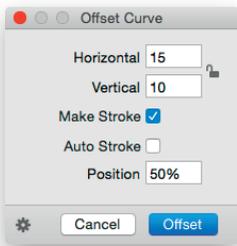
path as source. If there is no path, Glyphs will put a copy of the current path into the background. Alter the appearance of the hatching by changing the background path and applying the filter once again.



With the *Origin* option, you can change the position of the hatch strokes. The two fields represent the x and y values of the construction origin. The *Step Width* option determines the distance between individual strokes, and *Angle* their slant in degrees. The *Offset Path* value determines the thickness of the strokes, much like the *Offset Curve* filter would. Deactivate the checkbox next to it to prevent the offsetting, and only insert straight, open paths as hatch lines. This may be useful in case you plan additional subsequent transformations.



To apply *Hatch Outline* as a custom parameter, use the string `HatchOutlineFilter; OriginX:<x>; OriginY:<y>; StepWidth:<distance>; Angle:<angle>; Offset:<thickness>` for the *Value* of the custom parameter.



5.2.3 Offset Curve

Changes the thickness of stems horizontally and / or vertically. The lock sign uses the horizontal value for both horizontal and vertical expansion. Without the *Make Stroke* option, paths will just be moved into a parallel position:



With the *Make Stroke* option, selected paths will be expanded to closed outlines:



Tip: You can apply the *Make Stroke* option to an open path to get a thickened outline. This way, you can get an initial outline just by drawing the internal stroke.

The *Position* setting controls the distribution of the expansion. At 0%, the path will only expand to the right. At 100%, the path will only expand to the left. At 50%, the expansion will be evenly distributed to both sides of the path. Right and left sides are determined by the path orientation.

With the *Auto Stroke* option, the vertical dimensions will be kept intact. In that case, the offset position will be assumed at 50%.



To apply *Offset Curve* as a custom parameter, use *OffsetCurve*; <x>; <y>; <stroke>; <position/auto> for the *Value*, where <stroke> can be 1 for yes, and 0 for no, and <position> must be broken down to a floating point value between 0.0 (equivalent to 0%) and 1.0 (100%). If you want to use the *Auto Stroke* option in the parameter, use the string 'auto' as fourth argument.

5.2.4 Remove Overlap

The *Remove Overlap* filter has no dialog. When triggered, it immediately removes overlaps of selected paths, or all paths if none are selected. It also clears the selected glyph of all open paths and stray nodes. The filter expects all outline orientations to be set correctly (see 3.3.12, 'Controlling Path Direction', p. 25). To maintain overlaps in shapes while editing, you can have overlaps removed automatically at export with the respective option in the export dialog (see 15.4, 'Type 1, OpenType, and TrueType', p. 165). To apply this filter as a custom parameter, use *RemoveOverlap* as the *Value* without further arguments.

5.2.5 Roughen

Segments an outline into straight subpaths and randomly moves the nodes within a given limit. Control the size of the subpaths with the *Segment Length* field. The *Horizontal* and *Vertical* values control the maximum offset for each node. With the *Angle* value, you can tilt the horizontal and vertical node transformation. If you use modest values, the resulting glyphs will receive a roughened look, hence the name:

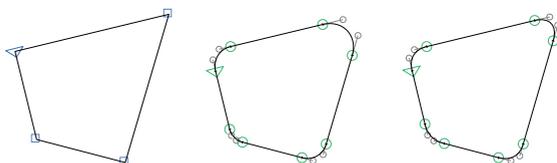


To apply this filter as a custom parameter, use *Roughenizer*; $\langle length \rangle$; $\langle x \rangle$; $\langle y \rangle$; $\langle angle \rangle$ for the *Value*.

5.2.6 Round Corners

Use this filter to round all selected corners of a path. To only round the outside corners, i.e., corners pointing into the white background, simply select nothing. Supply a *Radius* value in units. Use the *Visual Corrections* option to create a more natural looking corner rounding. This option increases the corner radius at obtuse angles, and reduces the radius at acute angles, yielding a more natural shape.

From left to right: original shape, rounded shape, and rounded shape with visual corrections.



To apply this filter as a custom parameter, use *RoundCorner*; $\langle radius \rangle$; $\langle correction \rangle$ for the *Value*. Use a negative $\langle radius \rangle$ for rounding (white) inside corners. The $\langle correction \rangle$ can either be 1 for yes or 0 for no.

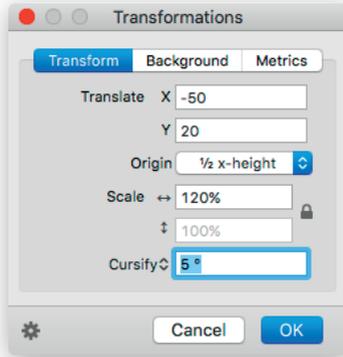
5.2.7 Rounded Font

The *Rounded Font* filter has no dialog. It reads the first vertical stem of the master, and tries to apply appropriate corner rounding with overshoots to the selected glyphs.

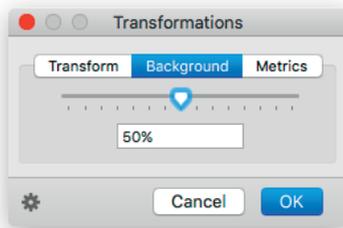
It can be triggered as a custom parameter with *RoundedFont* as the *Value* and appropriate stem settings in *File > Font Info > Masters*. Alternatively, you can supply a different stem value inside the parameter string: *RoundedFont; <verticalStem>*.

5.2.8 Transformations

The window of the Transformations filter sports three tabs: *Transform*, *Background*, and *Metrics*.

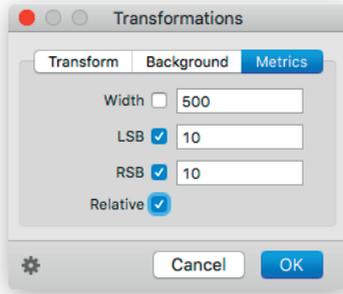


Use *Transform* to horizontally and vertically move, scale, and skew outlines. For scaling and skewing, you can set the transformation origin to cap height, half cap-height, x-height, half x-height, and baseline. Skew without optical correction by activating the *Slant* option, or with optical correction using the *Cursify* option. Cursify requires correctly set vertical and horizontal stems in *File > Font Info > Masters*.



Use the slider in the *Background* tab to interpolate between front and background paths. The paths need to be compatible. This is useful if you want to experiment with a feature of a glyph, e.g., the length of a stroke. You can quickly copy a

path into the background by choosing *Glyphs > Selection to Background* (Cmd-J).



In the *Metrics* tab, you can set the width and sidebearings of all selected letters at once. With the *Relative* option, the values will be added or subtracted.

The *Value* code for the custom parameter contains a range of optional arguments: *Transformations*; *LSB*:<±*/shift>; *RSB*:<±*/shift>; *Width*:<±shift>; *ScaleX*:<percent>; *ScaleY*:<percent>; *Slant*:<amount>; *SlantCorrection*:<bool>; *OffsetX*:<amount>; *OffsetY*:<amount>; *Origin*:<select>. The <shift> arguments optionally take the indicated prefixes for relative operations, or will set it to the specified value if no operators are supplied. The <percent> arguments are the actual percentage numbers, i.e., 100 means no scaling. For <amount>, you can use any number, positive or negative, with or without decimals, <bool> can be 1 for yes (*Cursify*) and 0 for no (*Slant*), and <select> is a number between 0 and 4, representing the five options of the transformation origin menu.

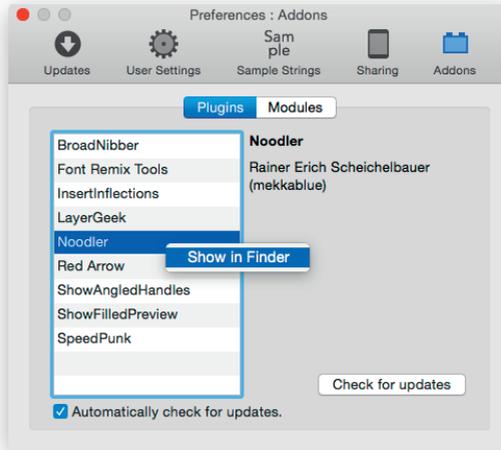
5.3 THIRD-PARTY FILTERS

Tip: You can find a range of third-party filters on glyphsapp.com/extend

You can extend the functionality of Glyphs by installing additional filters. You can do so by double-clicking the file carrying the *.glyphsFilter* suffix. Glyphs will automatically copy the filter file into the correct location, the *Plugins* subfolder of the app's *Application Support* folder. After a restart of the app, the filter will show up in the *Filter* menu. Refer to its documentation for further details.

You can uninstall a filter by removing the filter from the *Plugins* folder inside the *Application Support* folder of Glyphs. The quickest way to navigate there is to open *Glyphs > Preferences > Addons > Plugins* (Cmd-comma), then right-click

the name of a plugin, and choose *Show in Finder* from the upcoming context menu.



Many plugins can be installed, kept up to date automatically, and also uninstalled through the Plugin Manager, accessible via *Window > Plugin Manager*. For more information, see section 16.2.2, 'Plugin Manager' (p. 173).

6 Font View

6.1 VIEWING GLYPHS

The Font view of the main window displays the glyphs in the font. Here, you can manage your glyph set. When multiple tabs are open, you can activate the Font view by clicking on the first tab, or pressing Cmd-Opt-1. The Font tab has two viewing modes, grid view and list view.

6.1.1 Grid View



You can switch the Font view to grid mode by clicking the grid symbol in the double button in the top left corner of the window:



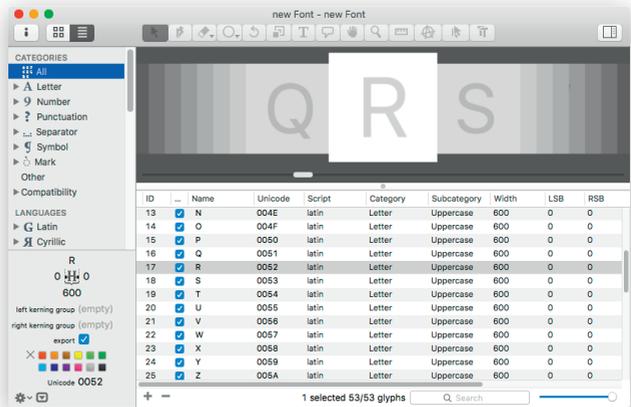
With the slider in the bottom right corner of the window, you can control the zoom level of the displayed glyphs. Depending on the zoom level, Glyphs will display either the glyph images only, or glyph images with glyph names, or, in large zoom stages, images, names and Unicode values, if available.

The glyph cell shows some extra information about the glyph. A yellow warning sign in their top right corner indicates that the glyph contains a metric key which is out of sync. See section 9.1.2, 'Metric Keys' (p.114) for more details. Glyphs are shown in their glyph color. Glyphs with unsaved changes show a slightly darker glyph cell until saved.

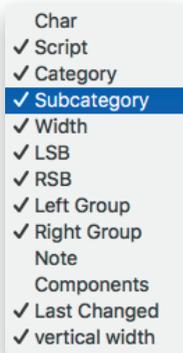


A red top-left corner indicates that not all the layers used for interpolation are compatible with each other. For more details on how to keep drawings compatible, see section 12.4, ‘Fix Outline Incompatibility’ (p.144).

6.1.2 List View



You can set the Font view to list mode by clicking the button with the list icon in the top left corner of the window:



In list mode, each column represents a glyph property. You can sort glyphs by any of their displayed properties if you click on the corresponding column header. Click again to reverse the sort order. Click and drag a header to rearrange columns. Via the context menu of the column headers, you can control which columns are displayed:

- *Char*: a representation of the Unicode character with the system font(s),
- *Script*: the script the glyph is associated with, spelled in lowercase, e.g., ‘latin’, ‘greek’, or ‘arabic’,
- *Category* and *Subcategory*: the categories, e.g., Letter, Number, or Symbol, and subcategories, e.g., Uppercase, Lowercase, or Space, as far as available,
- *Width*, *LSB*, and *RSB*: the horizontal metric values

- *Left Group* and *Right Group*: the kerning groups for the left and right sides of the glyphs,
- *Note*: an arbitrary text stored inside the glyph. You can search for it with the search field,
- *Components*: the component structure of compounds. You can change the components by typing a new glyph names in this field,
- *Last Changed*: the date of the last change made inside the respective glyphs,
- *Vertical Width*: the vertical width, necessary for vertical scripts like Mongolian.

6.1.3 Searching for Glyphs

You can choose to view different excerpts of your glyph set by selecting categories, languages or filters from the left sidebar. To further narrow down your search, you can enter a search term in the search field (Cmd-F) at the bottom right of the window. For more details, see section 6.5.1, 'Search Box' (p. 74)

The three numbers to the left of the search field indicate (from left to right) how many glyphs are selected, how many are currently displayed, and the total number of glyphs in the font file:



If you cannot see a glyph you are looking for, the current display settings may prevent the glyph from showing. If that is the case, first check if there is a selection in the left sidebar, or simply click on *All* at the top. Also, a search term may still be entered and active in the search field. Click in the search field (Cmd-F), and clear its search string.

6.2 MANAGING THE GLYPH SET

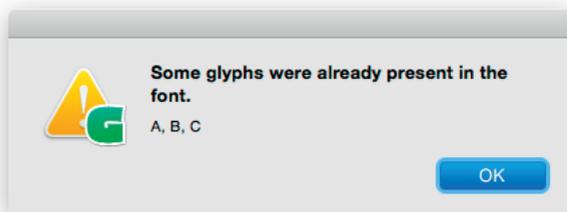
6.2.1 Generating New Glyphs

There are several ways to add new glyphs to a font:

- *Glyph > New Glyph* (Cmd-Opt-Shift-N) and the Plus button at the bottom in Font view add an empty glyph called 'newGlyph' to the font. You may need to scroll down to see it.

Adding glyphs through menu commands also works in Edit view. Newly added glyphs will then be added to the sample string in the current tab.

- *Glyph > Duplicate Glyph* (Cmd-D) duplicates the currently selected glyph(s). Because glyph names must be unique throughout a font, the new glyphs will carry an incremental number as name suffix, e.g., ‘.oo1’, ‘.oo2’, etc.
- *Glyph > Add Glyphs* (Cmd-Shift-G) opens a dialog where you can insert a list of whitespace-separated glyph names. Pressing the *Generate* button will add all of these glyphs, provided they are not in the font yet. You can use custom recipes for building compound glyphs. For a detailed description of component recipes, section 8.1.3, ‘Recipes’ (p. 99). To add all glyphs corresponding to a range of Unicode characters, type the names of the first and the last glyph, separated by a colon, e.g., ‘uniE000:uniE0FF’. Alternatively, you can simply type the characters themselves, e.g., ‘Ä:ñ’ or ‘あ:け’. Glyphs already available inside the font will be ignored, and you will be notified that at least some of the glyphs were already there:



For the complete list of recognized glyph names, choose *Window > Glyphs Info*.

- Open *Window > Glyph Info*, then select one or more glyph descriptions, and click the *Add to Font* button.
- Some entries in the sidebar of the Font view have a number badge, indicating the number of glyphs already created versus the total number of glyphs predefined in that category, language group or filter. Right-click or Ctrl-click the badge to get a list of all glyphs missing in this category.

Select one or more glyph names, or press Cmd-A to select all of them, and click *Generate* to add them to your font:



6.2.2 Copying Glyphs Between Files

You can copy any number of glyphs from one file (the source) and paste them into another file (the target). Components will be re-linked to their respective counterparts in the target font. If a glyph with the same name already exists in the target font, then the pasted glyphs will be renamed with an increasing triple-digit suffix, e.g., 'A.001', 'A.002' etc., avoiding data loss.

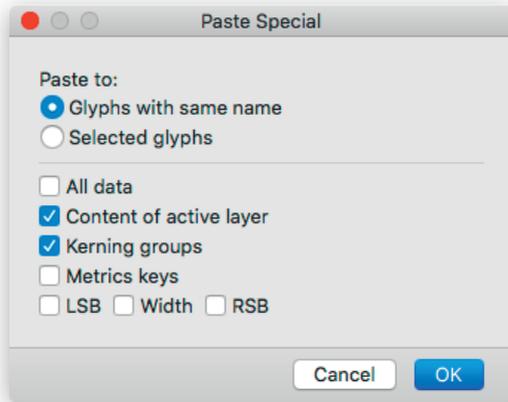
If you want to overwrite existing glyphs, you can use the *Paste Special* (Opt-*Cmd-V*) command, which appears in the *Edit* menu when you hold down the Option key. A dialog will ask you what exactly you want to do. First, you are asked about the target of the Paste operation:

- *Glyphs with the same name* looks for glyphs with the same glyph names as the ones pasted and will replace them if present, or add them if not.
- *Selected glyphs* will keep the target glyph names but replace their contents, including their layer structure. If the number of selected glyphs differs between copying and pasting, it will quietly ignore additionally selected target glyphs or additionally copied source glyphs.

Then, you are asked about what exactly of the copied data you want to paste into the target glyphs:

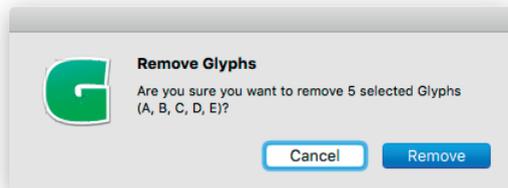
- *All data* disables all other options, as everything will be copied: paths, anchors, components, layers, glyph and layer attributes, and metrics.
- *Content of active layer* acts like the previous option, but it ignores all layers except for the one currently displayed.

- *Kerning groups* only copies the left and right kerning groups. This is useful for reduplicating kerning between similar fonts.
- *Metrics Keys* copies only the sidebearing and width formulas for recalculation. Afterwards, you may need to update the keys with *Glyph > Update Metrics* (Ctrl-Command-M).
- The options *LSB*, *Width*, and *RSB* copy the respective metric values into the target glyph(s). If you select both *Width* and *RSB*, the value of the right sidebearing will take precedence over the width value.



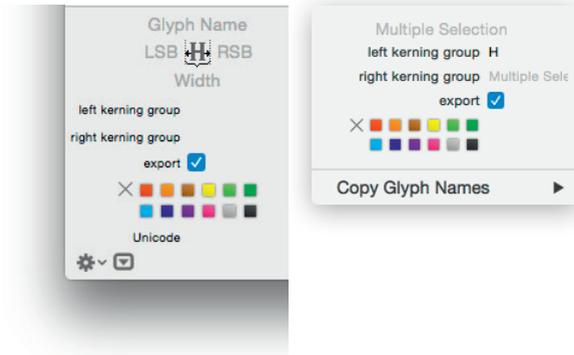
6.2.3 Removing Glyphs

Select any number of glyphs in Font or Edit view, and choose *Glyph > Remove Glyph* (Cmd-Backspace or Cmd-Delete). This also works with the current glyph in Edit view. In Font view, you can also click the Minus button in the bottom section of the window. Before the respective glyphs are removed from your font, you will be asked for confirmation:



6.3 GLYPH PROPERTIES

Select one or more glyphs and, if necessary, click on the second button  at the bottom left of the window, and the glyph properties will be displayed. Some of the properties can also be accessed through the context menu of one or more selected glyphs.



In List view, glyph properties are displayed and can be edited in columns. Some properties are only displayed in the columns of List view. Right-click on the column headers and tick off the columns you want to see. You can use them as sort key. Switch between ascending and descending sorting by clicking on the column title.

6.3.1 Name

The glyph name can be accessed and changed only if exactly one glyph is selected. If you want to batch-change the names of many selected glyphs, you can use *Edit > Find > Find and Replace* (Cmd-Shift-F). See section 6.6.3, 'Renaming Glyphs' (p. 82) for more details.

6.3.2 Width and Sidebearings

The widths as well as the left and right sidebearings can be accessed and changed for any number of selected glyphs at once. Changing only the width affects the right sidebearing.

6.3.3 Kerning Groups

You can set kerning groups for a range of glyphs, see section 9, 'Spacing and Kerning' (p. 114), for further details.

6.3.4 Exports

The glyph will appear in the exported font only if this check box is enabled. This is useful for keeping glyph variations or glyph components from ending up in the final font. Newly created glyphs are set to export by default, unless their name starts with an underscore ('_').

6.3.5 Color Label

For easier sorting and filtering, or simply for keeping oversight, you can mark glyphs with one of twelve predefined colors. The colors have no effect on the exported font.

By holding down the Option key, you can set colors for the currently displayed layers rather than for the whole glyphs. In that case, the glyph-wide colors will be shown on the left halves of the glyph cells, while the layer-specific colors occupy the right halves of the cells.

From left to right:
orange glyph color (a); orange
glyph and pink layer color
(b); no glyph color and pink
layer color (c); neither glyph
nor layer color set (d).



6.3.6 Unicode

The Unicode value is determined by the glyph name, which means that you cannot set the value directly. Thus, the displayed Unicode value of glyphs is read-only. However, if the *Use Custom Naming* option is activated in *File > Font Info > Other Settings* (Cmd-I), you can manually set the Unicode value of any glyph.

Glyphs in the Private Use Area can always have custom names. You first create them with a name according to the uniXXXX (for values between U+0000 and U+FFFF) or uXXXXX scheme (for values beyond U+10000) to set their Unicode value, and then rename them to your liking. The usual glyph naming restrictions still apply, though (see 6.6.2, 'Naming Glyphs', p. 80). Make sure the custom name you choose is not already reserved by another glyph in the internal glyph database. Otherwise, the Unicode value of that specific entry will be applied. In that case, you can override the glyph data through *Edit > Info for Selection* (Cmd-Opt-I). For details, see section 17.4.2, 'Local Glyph Data Changes' (p. 206).

6.3.7 Note (List View)

A glyph can have an arbitrary string as note. Notes are only accessible in List view. You can search for glyph notes in the search field of the Font tab.

6.3.8 Components (List View)

In List view, you can have Glyphs display the elements of composite glyphs by right-clicking on the table header and activating *Components* in the context menu. You can change the composition of glyphs by changing their entries in that column, e.g., from ‘A, macron’ to ‘A, macron.case’.

6.3.9 Read-Only Properties in List View

The following properties can only be accessed in List view. They are read-only and only useful for sorting glyphs:

- *ID* corresponds to the glyph order in the font. You can influence the order by setting the custom parameter *glyphOrder* in *File > Font Info > Font*.
- *Character*, i.e., the Unicode character as represented by the OS X system font or a fallback font. Only glyphs that have a Unicode value assigned can display something in this column.
- *Script*, e.g., ‘Latin’, ‘Greek’, ‘Cyrillic’, this is set through the glyph name.
- *Category* and *Subcategory*, e.g., ‘Letter’, ‘Number’, and ‘Uppercase’, respectively. This is set through the glyph name.
- *Last Changed*, i.e., date and time of the last manipulation of the respective glyph.

6.4 BATCH-PROCESSING

6.4.1 Selecting Glyphs

Batch-processing a number of glyphs requires a selection of multiple glyphs. You can select all glyphs currently displayed in Font View via *Edit > Select All* (Cmd-A). Cancel the glyph selection with *Edit > Deselect All* (Cmd-Opt-A). To select all displayed glyphs except some of them, first select the ones you want to exclude, and then choose *Edit > Invert Selection* (Cmd-Opt-Shift-I).

6.4.2 Menu Commands

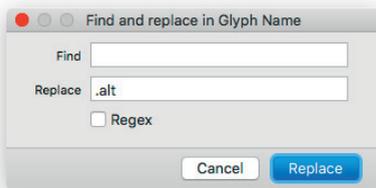
You can apply manipulations on the active layers of a range of selected glyphs. Select the glyphs you want to batch-edit, and choose one of these commands:

- *Glyph > Duplicate Glyph* (Cmd-D)
- *Glyph > Update Glyph Info*
- *Glyph > Make Component Glyph* (Cmd-Opt-Shift-C)
- *Glyph > Decompose Components* (Cmd-Shift-D)
- *Glyph > Add Image* (see 6.7, 'Images', p. 82)
- *Glyph > Update Metrics* (Ctrl-Cmd-M) and *Update Metrics for All Masters* (Ctrl-Opt-Cmd-M)
- *Glyph > Set Anchors* (Cmd-U) and *Reset Anchors* (Opt-Cmd-U)
- *Paths > Selection to Background* (Cmd-J), *Add Selection to Background* (Cmd-Opt-J), *Clear Background, Assign Background, Swap with Background* (Ctrl-Cmd-J)
- *Paths > Correct Path Direction* (Cmd-Shift-R) and *Correct Path Direction for All Masters* (Cmd-Opt-Shift-R)
- *Paths > Round Coordinates*
- *Paths > Tidy up Paths* (Cmd-Opt-Shift-T)
- *Paths > Add Extremes*

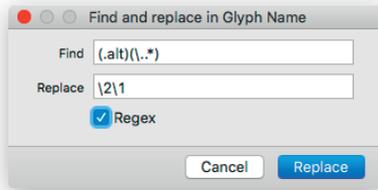
6.4.3 Batch-Renaming Glyphs

You can search and replace in names of selected glyphs with *Edit > Find > Find and Replace* (Cmd-Shift-F). It will look for the string entered in the *Find* field and replace it with the string entered in *Replace*.

You can add characters to the end of the names of selected glyphs if you leave the *Find* field empty and enter the desired suffix in the *Replace* field:



The *Regex* option enables regular expressions in both text entry fields. The text in the entry fields will be displayed in red as long as the regular expression does not validate.



For instance, you can use a regular expression if you have a number of glyphs with multiple dot suffixes, but want to move 'alt' to the end. Then you would enter '\.alt)(\..*)' in *Find*, and '\2\1' in *Replace* and press the *Replace* button. '.' stands for a literal dot, '*' means any number of any characters; '\1' represents the first set of parentheses, '\2' the second one. For more examples, see section 6.5.1, 'Search Box' (p. 74).

6.4.4 Filters

All built-in, and most available third-party filters can be applied to more than one selected glyph. For further details, refer to their documentation.

6.4.5 Palette Manipulations

From the Palette (Cmd-Shift-P), you can apply the mirroring, scaling, rotating, and skewing functions in the Transformations section on multiple selected glyphs. The transformation origin will be respected.

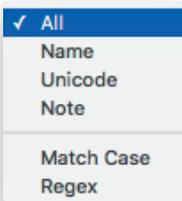
Also, the boolean operations work with a multiple-glyph selection. Be careful though, it is easy to achieve an unexpected result.

6.5 FILTERING AND SORTING

6.5.1 Search Box

The search box (Cmd-F) at the bottom right provides a way to instantly narrow down the selection of glyphs displayed in the Font tab. By clicking on the search symbol (the magnifying glass) in the entry field, you can choose the following options:

- *All*: search in glyph names, glyph notes, and Unicode values
- *Name*: search in names only



- *Unicode*: search in Unicode values only
- *Note*: search in glyph notes only
- *Match Case*: if active, searches will be case-sensitive
- *Regex*: if active, the search string will be interpreted as a regular expression

The *Regex* option not only activates regular expressions, but also automatically limits the search to glyph names. Regular expressions allow you to search for a text pattern rather than a specific text. When activated, you can use search terms like `[Lldt]caron(\..+)*` to match Lcaron, lcaron, dcaron, tcaron, with or without dot suffixes. Some examples:

- `[abc]` any of the letters *a*, *b* or *c*
- `[^abc]` any letter but *a*, *b* or *c*: *d*, *e*, *f*, *g*...
- `.` any character: *a*, *b*, *c*, *.*, *-*, *1*, *2*, *3*...
- `\d` any digit: *0*, *1*, *2*, *3*, *4*, *5*, *6*, *7*, *8*, or *9*
- `\.` a literal period: *.*
- `a?` zero or one occurrence of *a*
- `a+` one or more occurrence of *a*: *a*, *aa*, *aaa*, *aaaa*...
- `a*` zero, one or more occurrences of *a*
- `a{3}` exactly three occurrences of *a*: *aaa*
- `a{2,5}` between 2 and 5 occurrences of *a*: *aa*, *aaa*, *aaaa*, *aaaaa*
- `a.*` any glyph name with *a* at the beginning: *a*, *ae*, *aacute*...
- `*a` any glyph name with *a* at the end: *a*, *cedilla*, *ccedilla*...
- `+-.+` any glyph name with a hyphen in it: *beh-ar*, *he-hb*...
- `+\..+` any glyph name with a period in it: *a.sso1*, *J.titl*...
- `+\d` any glyph name with a digit at the end: *a.001*, *b.x2*, *c.sso2*
- `[^\d]*\d{3}` any glyph name with exactly three digits at the end, and no digits anywhere before: *a.001*, *x.test123*...

6.5.2 Categories

Categories give you the option to only display a predefined subset of the glyphs you have. Click the triangle next to the category name to access subcategories. You can combine more than one category by Cmd-clicking their names, e.g., if you Cmd-click the categories *Uppercase* and *Lowercase*, Glyphs will display both upper- and lowercase letters.

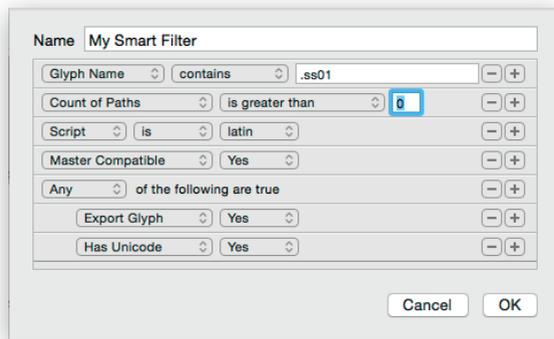
You can achieve an even narrower subset by also Cmd-clicking a Filter. For instance, if you click the *Uppercase* category and then Cmd-click *MacRoman*, Glyphs will display all uppercase characters in the MacRoman encoding.

You can cancel any subset you created and return to displaying the complete glyph set by clicking on the *All* category.

6.5.3 Languages

To only display glyphs belonging to a certain script, you can click on the name of the script in the Languages, e.g., *Latin*, *Cyrillic*, *Greek* etc. Clicking on the triangle next to the script name will expand the view and give you more detailed subsetting options. Again, you can combine subsets by subsequently Cmd-clicking them.

6.5.4 Smart Filters



To define a query-based filter, click the gear button in the bottom left of the window and choose *Add Smart Filter*. Smart filters are similar to smart playlists in iTunes. Set up the query with any of these properties:

- *Category* and *Subcategory*: the categorization of the glyph according to the glyph info in effect for the font.
- *Color Label*: the color assigned to the glyph, if any.
- *Count of Paths*: number of paths in the first master.
- *Export Glyph*: whether the glyph exports or not.
- *First Master Has Components*: whether the first master contains components or not.
- *Glyph Name*: the name of the glyph.
- *Has Annotations*: whether the glyph has annotations, placed with the Annotations Tool (A).
- *Has Component*: whether the first master contains a component with a specific name.

- *Has Custom Glyph Info*: whether the glyph has custom glyph info applied (via *Edit > Info for Selection*) that deviates from the default glyph info database.
- *Has Hints*: whether any manually placed hints are on any of the layers in the glyph.
- *Has Metrics Keys*: whether any of the glyph's metrics contains a formula such as ' $n+10$ '.
- *Has Special Layers*: whether the glyph has a Bracket or Brace layers.
- *Has Unicode*: whether the glyph has a Unicode value.
- *Is Auto-Aligned*: whether the glyph is automatically aligned.
- *Master Compatible*: whether masters can interpolate.
- *Metrics Keys Out of Sync*: whether the metrics keys for the glyph are up to date or not.
- *Script*: e.g., Latin or Arabic according to the glyph info.

Add additional properties by clicking on the plus button at the end of a line. By default, all properties are applied to the query (logical-and). You can add and nest logical-and ('all of the following'), logical-or ('any of the following') and even logical-not ('none of the following') concatenations by holding down the Option key while pressing the plus button.

6.5.5 List Filters

You can define a list of glyphs to be displayed by choosing *Add List Filter* from the gear menu in the bottom left of the Font tab. By default, currently selected glyphs are added to the list automatically. You can subsequently edit the list or paste a list of glyph names. In Font view, once you apply a list filter, you will see its glyphs in the order they are listed in.

6.5.6 Manage Filters

Once you have set up a couple of filters, you can apply more than one filter by holding down the Command key and subsequently clicking the individual Filter names.

Edit an existing filter by selecting it in the Font tab sidebar and subsequently choosing *Edit Filter* from the gear menu in the bottom left. Alternatively, double click its side bar entry. A dialog sheet will appear that allows you to edit the selected smart or list filter. Remove an existing filter by selecting its sidebar entry and choosing *Remove Filter* from the same gear menu.

You can sort your filters by dragging them into a desired order. You can also organise your filters in sidebar folders. To add a folder, choose *Add Folder* from the gear menu.

To quickly change the name of a filter or folder, click to select it, press the Return key, and type a new name.

6.5.7 Custom Parameter ‘glyphOrder’

Categories and Filters have no influence on the order of the glyphs in the final font file. You can take control of the saving order with the ‘glyphOrder’ custom parameter in *File > Font Info > Font* (Cmd-I). The parameter takes a list of glyph names as its value. If Glyphs finds this custom parameter, it will also display the glyphs in Font view in that order. When importing an existing OTF or TTF, Glyphs preserves the glyph order and creates the custom parameter automatically if the preference *Keep Glyph Names from Imported Files* is activated (see 2.1.2, ‘User Settings’, p. 12).

6.6 NAMES AND UNICODE

6.6.1 Glyph Info Database

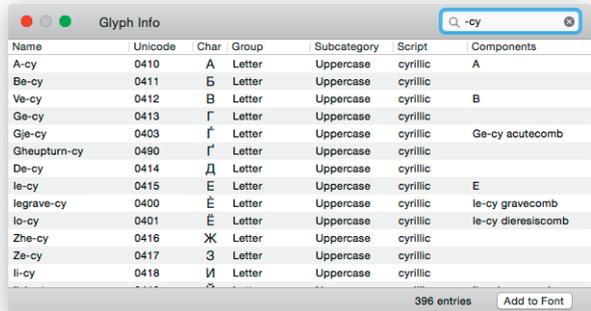
Glyphs contains a glyph info database. For each glyph, the database defines:

- the human-readable (or ‘nice’) glyph name,
- the production names, i.e., the name as required for the exported installable font file,
- the associated Unicode value, if any,
- possible components,
- default anchors, if any,
- possible accents, mainly for displaying the mark cloud when you click on an anchor,
- the script such as Latin, Cyrillic, Arabic, etc.,
- the category and subcategory, such as letter and uppercase, punctuation, or figure, etc.

For the ‘nice names’, Glyphs employs a special naming convention which is, for the most part, loosely based on the Adobe Glyph List (AGL) Specification. But in the AGL, not all glyphs have nice names. Rather, they have names like ‘uni042F’, which are hard to memorize. In cases like this, Glyphs derives the nice name from the Unicode description and appends a script suffix, e.g., ‘CYRILLIC CAPITAL LETTER A’ becomes ‘A-cy’. The glyph info database is stored as an XML

For more information about the Adobe Glyph List, see github.com/adobe-type-tools/agl-aglfn

file inside the application. You can extend or override the internal database with a custom XML file in the Application Support folder for Glyphs. For details, see section 17.4.1, ‘Global Glyph Data Changes’ (p.204) in the Appendix. After a restart, Glyphs will respect the custom changes of your XML for all files produced thereafter. For an overview of the available data stored in the glyph info database, choose *Window > Glyph Info*:



You can override this default or custom glyph info locally, i.e., for a single Glyphs file. Select a glyph, or a range of glyphs, and choose *Edit > Info for Selection*. A dialog sheet will appear, presenting option for manipulating the glyph info. This is useful if you need to have different glyph options for one project.



For more info on this, see section 17.4.2, ‘Local Glyph Data Changes’ (p.206) in the Appendix.

Typically, suffixed glyphs will inherit their glyph info attributes from their unsuffixed counterpart. That means that, e.g., both ‘X’ and ‘X.alt.bold’ will be treated as letters, thus

enable automatic alignment and receive default anchors when *Glyphs > Add Anchors* is run. If you name the copy ‘Xalternate’ without the dot, it will not be recognized as a letter unless you changed the glyph data accordingly. So, if you want to handle glyph backups and copies in a way that Glyphs can still semantically connect them with their originals, make sure the glyph names are the same up until the first period in the name.

6.6.2 Naming Glyphs

The glyph name appears below the glyph in grid view, or in a separate column in List view. You can edit a glyph name by clicking once into the name. Glyphs will automatically convert entries like ‘ä’ or ‘uni00E4’ to nice names, i.e., ‘adieresis’ in this case. Based on the glyph names, Glyphs will automatically assign Unicode values as well as Category, Subcategory and Script attributes, and can even automatically build some OpenType features.

When you name or rename a glyph, three different types of input values are accepted:

- the ‘nice’ name, e.g., ‘la-cy’
- the Unicode value in hex form with a ‘uni’ or ‘u’ prefix, e.g., ‘uni042F’ or ‘u10120’
- the Unicode character as typed with the keyboard, e.g., ‘Œ’

This automation can be deactivated on a per-font basis:

Activate File > Font Info > Other Settings > Use Custom Naming.

If you do so, however, other automatic functions, like the composition of compound glyphs or the generation of OpenType features, will stop working.

For Unicode-value based glyph names, use the prefix ‘uni’ followed by the four-digit hexadecimal code, e.g., ‘uniE002’. For Unicodes outside the Basic Multilingual Pane (BMP), use ‘u’ as a prefix, followed by the five- or six-digit code, e.g., ‘u10015’.

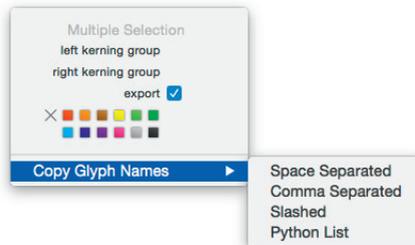
Except for ‘notdef’, a valid glyph name must begin with a character from A-Z or a-z. A glyph name can subsequently also contain figures (0-9), underscores (_), periods (.) or hyphens (-). All other characters, including whitespace characters, are not allowed. A dialog will appear if you try to use invalid characters like space in a glyph name.

You can use preceding underscores for non-exporting glyphs. When the name of a newly generated glyph starts with an underscore, its *Exports* option will be off by default.

If you need variations of glyphs, extend your glyph names with dot suffixes, like 'n.sc' for a small cap n, or 'five.sups' for a superscript five. Some suffixes will be used by Glyphs to automatically build OpenType feature code. See the appendix of this manual for a list of recognized suffixes. Multiple suffixes should be added in the order of the OpenType features, that is, if you want to take advantage of the automatic feature generation. E.g., if you want to add a stylistic alternate for a small cap c, then you call it 'c.sc.sso1', since, by default, stylistic sets come after small caps.

Hyphens are used to indicate the script a glyph belongs to, e.g., 'alef-ar' for an Arabic alef or 'ta-hira' for the ta syllable in the Japanese Hiragana script.

To name ligatures, the names of the parts need to be joined with an underscore, e.g., 'f_l' for an ffl ligature. The script suffix is appended only once (e.g., 'lam_alef-ar'). Variation suffixes are only added once as well. These suffixes define the role of the whole ligature. For instance, a ligature formed of 'lam-ar.init' and 'alef-ar.medi' is called 'lam_alef-ar.init'.



You can copy the names of selected glyphs into the clipboard by opening the submenu *Copy Glyph Names* in the context menu. You can choose between four options:

- *Space Separated*: The names will be copied with simple spaces between them, e.g., 'a adieresis aacute b c'. This is useful for building OpenType feature code.
- *Comma Separated*: Like above except, e.g., 'a, adieresis, aacute, b, c'. This is useful for building custom parameter strings. It is also the most human-readable format, so you may want to use this in written communication with collaborators.

- *Slashed*: The names will be copied with forward slashes as delimiters, e.g., `'/a/adieresis/aacute/b/c'`. This is useful for building and exchanging sample strings, especially if you need to include glyphs that have no Unicode values. When pasted into an Edit tab, Glyphs will try to parse slash-escaped glyph names.
- *Python List*: A Python-style list of strings will be copied in the clipboard, e.g., `["a", "adieresis", "aacute", "b", "c"]`, ready for pasting into Python code.

6.6.3 Renaming Glyphs

You can rename any glyph by simply clicking in its name field and editing the name as desired. Keep in mind that if you change the name before the first period, its glyph info attribution will be changed as well. This means that Unicode value, category and subcategory, script, associated default anchors and accents, as well as the default decomposition may change.

For searching and replacing in glyph names, or batch-renaming a large number of glyphs at once, see section 6.4.3, ‘Batch-Renaming Glyphs’ (p. 73).

6.6.4 CID Mapping

CJK fonts make use of CID mapping, where glyphs are accessed not by their glyph name, but through a unique character identifier (CID). Glyphs supplies mapping files for its ‘nice’ glyph names to CIDs. A ROS determines which glyphs receive which CID. Glyphs that are present in the font and set to export, but not included in the ROS, are added to the end of the CID mapping at export. For more information about how to choose a ROS, see its entry in the list of custom parameters in the Appendix (p.195).

6.7 IMAGES

6.7.1 Adding and Managing Images

Similar to Edit view, you can add an image to the current glyph layer by dragging it onto a glyph cell. Batch-add images by choosing *Glyph > Add Image* and then selecting any number of images. Glyphs will place them in the appropriate cells based on their file name. E.g., an image called `ntilde.png` will be put in `ntilde`, `iacute.pdf` in `iacute`.

6.7.2 Viewing Images

Like in Edit view, image display in Font view respects the *View > Show Image* setting unless there is no path drawn in the currently active layer of the glyph. In that case, the image will be shown regardless of the *Show Image* setting.

For more details on what you can do with images, see section 3.11, 'Images' (p. 44).

7 Font Info

Open the Font Info window by choosing *File > Font Info* (Cmd-I), or by clicking on the Info button in the top left corner of the main window.

7.1 FONT

The Font tab contains information that applies to the whole font family.

7.1.1 Family Name

The name of the font family as it will appear in a font menu. Fonts carrying the same family name will be grouped in the same Style submenu. You can use a space in the font name, but non-ASCII characters may prevent the font from exporting. If you need special characters in your family name, consider the custom parameter *Localized Family Name*.

Glyphs uses the *Family Name* entry to derive the file name, and the entries for OpenType Name IDs 1, 3, 4, and 6. In CFF-based OpenType fonts, the entry is also used for *FontName* and *FullName* in the CFF table. The entry can be overridden by the *familyName* parameter in an instance.

Name IDs refer to the entries of the OpenType Naming Table stored in OTF and TTF fonts. For a complete specification, see: www.microsoft.com/typography/otspec/name.htm

7.1.2 Units per Em

Number of units per em square (UPM), 1000 is the default. Increasing the UPM value can improve the representation of subtle details. The OpenType specification allows values between 16 and 16,384, but values greater than 5000 can lead to problems in InDesign and Illustrator. Problems have been reported for other applications starting at 3000 UPM. Some applications expect a UPM value of 1000 for CFF fonts. Also, point coordinate values must not exceed $\pm 32,768$, and glyph widths of CFF fonts can be problematic beyond ± 4096 . Thus, if you need higher precision, it may be better to adapt the *Grid Spacing* and *Subdivision* values in *File > Font Info > Other Settings*. See section 7.5.1, 'Grid Spacing and Subdivision' (p. 96), for more details.

Click on the double arrow next to the text field to scale the entire font. If you want to enlarge the font, set the UPM to something smaller than 1000, then scale back to 1000. Start with a higher value than 1000 to scale down. The quotient of these two values determines the scale factor.

Tip: to avoid early rounding errors, it may be a better idea to use the Scale to UPM parameter in an instance.

7.1.3 Designer and Designer URL

Here, you can enter your name and a URL (including the protocol, e.g., `http://` or `ftp://`), for example:

- *Designer*: Jessica Doe
- *Designer URL*: `http://www.example.com/`

The entries correspond to OpenType Name IDs 9 and 12. You can check if the URL was entered correctly by clicking on the arrow next to the text entry field. Glyphs will open the URL in your web browser.

7.1.4 Manufacturer and Manufacturer URL

Here, you can enter name and URL (including the protocol, e.g., `http://` or `ftp://`) of your font vendor, for example:

- *Manufacturer*: Sample Font Store
- *Manufacturer URL*: `http://www.example.com/`

The entries correspond to OpenType Name IDs 8 and 11. You can check if the URL was entered correctly by clicking on the arrow next to the text entry field. The URL will then be opened in your web browser.

7.1.5 Copyright

A simple copyright notice. Click the circled arrow next to it to have Glyphs fill it in automatically, based on the entry in the *Designer* field. This will be recorded as Name ID 0.

7.1.6 Version

Glyphs derives the Version String (Name ID 5) from the *Version* entry. In the resulting font file, tools like `makeotf` and `ttfautohint` also leave their traces in the version string, e.g., `'Version 1.010;PS 001.010;hotconv 1.0.70;makeotf.lib2.5.58329'`. If you are not happy with this additional text, and therefore want to override the creation of the version string, then consider using the custom parameter *versionString* in *File > Font Info > Font*. See its entry in the list of custom parameters in the Appendix (p.201) for more details.

This additional information, however, may be helpful for renderers. There is a recorded case of an old version of `makeotf` creating a faulty glyph substitution table. Based on the extra data stored in the version string, InDesign recognizes those fonts, and reacts accordingly. Consider this before you remove the `makeotf` version info.

7.1.7 Date

For more details about the head table, see: www.microsoft.com/typography/otspec/head.htm

The creation date of the font. This entry will set the Creation and Modification dates in the OpenType Font Header table, also known as head table.

7.1.8 Custom Parameters

Use entries in the *Custom Parameters* field to specify more settings. To add a parameter, click on the plus symbol, select or type a *Property*, and enter a *Value*. Depending on the entry, you either type it directly into the field, or in a dialog that appears after clicking or tabbing into the field. You can also copy and paste parameters. You can use font-level custom parameters to override default values set by Glyphs, but they are themselves overridden by master and instance parameters.

Available parameters are displayed in the drop-down list that appears when you click on a *Property*. See section 17.3, ‘Custom Parameters’ (p.180) for a detailed description of each parameter.

7.2 MASTERS

While information about the designed masters (the input) is set under *Masters*, the *Instances* tab contains information about each font instance that will be generated (the output) when you run *File > Export* (Cmd-E). Clicking the plus button in the bottom left reveals a drop-down list with three options:

- *Add Master* adds a new, empty master with default values to the setup, i.e., Weight 100, Width 100, Custom 0. All glyphs will receive a new, empty master layer.
- *Add Other Font* asks you for a master in a file currently open in Glyphs; once you confirm the dialog, the chosen font master will be inserted in the front-most file. All glyphs will receive a new master layer containing whatever was in the corresponding glyph of the imported master, or empty if the glyph does not exist in the imported font master.
- *Duplicate Selected* creates an exact duplicate of the currently selected master in the same file. All glyphs will receive a new master layer with the same contents as the source master layer, including kerning.

Once you have several masters set up in the *Masters* tab, you can reorder them by dragging and dropping. The first master has special relevance for glyph-level hinting information. See section 10.4, ‘Manual hinting’ (p.126) and section 11.3,

For a step-by-step guide through setting up masters, see glyphsapp.com/tutorials/multiple-masters-part-1-setting-up-masters

Tip: A quicker way to duplicate a font master is to Opt-drag its entry in the sidebar to a new position.

'Manual Instructions' (p.133) for further details. Also, some smart filters only check for data in the first master. For more details, see section 6.5.4, 'Smart Filters' (p.76). All values will be interpolated if entered in the same order throughout all masters. You can edit several masters at once after you Shift-click or Cmd-click the master names in the list on the left.

7.2.1 Proportions: Weight, Width, and Custom

The *Weight* and *Width* pop-ups and the *Custom* text field affect only the toolbar icons and the naming of the master layers in the *Layers* palette in the Palette sidebar (Cmd-Opt-P). The style name of the final font and its interpolation are set in the *Instances* tab. The number value fields are important for interpolation. If you want the masters to export as they are, i.e., without any interpolating, the instances must carry the same interpolation values. For more information about this, see section 12.2, 'Setting up Masters' (p.142).

Proportions	
Weight	Light 45
Width	Regular 100
Custom	0
Vertical Stems	45
Horizontal Stems	41

7.2.2 Metrics

Here, you can enter the values for the vertical metrics of your type design:

- *Ascender*: the height of tall lowercase letters such as b, d, f, h, k, l or p.
- *Cap Height*: the height of the capital letters, not counting the overshoot.
- *x-Height*: the height of the small lowercase letters, i.e., without ascenders, not counting the overshoot.
- *Descender*: the depth of descenders of lowercase letters that reach below the baseline, e.g., g, j, p, or q.

These measurements serve as master-wide guidelines in Edit view when *View > Show Metrics* is active. When determining these values, ignore the overshoot in your design. So, for instance, if you have the choice between various values for

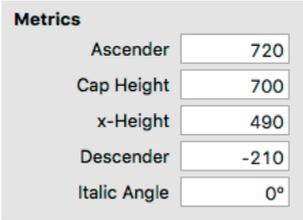
the x-height, say 490, 496 and 502, you want the one closest to your baseline, i.e., 490.

The vertical metrics *Ascender* and *Descender* have an impact on the line height as displayed in the Edit view. You can override the calculation and specify your own line height with the *EditView Line Height* parameter, either in the *Font* or *Masters* tab.

Glyphs will calculate the vertical metrics in the *OS/2* and *hhea* tables from these values. If you keep *Ascender* and *Descender* the same in all masters, the phenomenon of ‘line height jumping’ will be reduced to a minimum when switching between fonts in most applications.

If the *smallCapHeight* custom parameter is set, Glyphs will prefer its value to the *x-Height* value for the metric line when small cap letters are displayed. The same goes for Indic, Hebrew, and Arabic letters if the *shoulderHeight* parameter is set. Also, Glyphs is able to find alignment zones (see 7.2.4, ‘Alignment Zones’, p. 89) automatically if you first enter all vertical metrics correctly.

The *Italic Angle* has an effect on a number of items throughout the application. Functions that respect the slant angle include aligning anchors between two selected nodes, the display of the x offset in measurement mode, and the calculation of the sidebearings.



Metrics	
Ascender	720
Cap Height	700
x-Height	490
Descender	-210
Italic Angle	0°

7.2.3 Stems

If you enter good values for your standard stems, the autohinter will find those stems in your letters and put a stem hint on them. In a Multiple Masters setup, make sure the stems are listed in the same order in all masters, otherwise they cannot be interpolated correctly. For a detailed discussion, see chapter 10, ‘PostScript Hinting’ (p. 121).

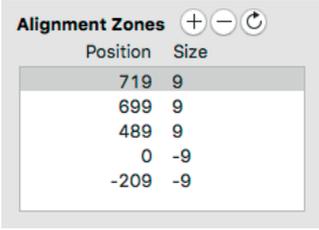
7.2.4 Alignment Zones

Alignment zones help create an even vertical alignment at low resolutions through overshoot suppression. The values entered here are crucial for the autohinting process when the font is being exported. An alignment zone must encompass anything that should later be aligned at a low resolution. For a detailed discussion, see chapter 10, 'PostScript Hinting' (p. 121).

After you have set the *Metrics* properly, you can click the gray circle to let Glyphs find the alignment zones for you. Glyphs will then reduplicate the heights of the vertical metrics you entered in the Metrics field as the positions of the zones. It also respects the *smallCapHeight* and *shoulderHeight* custom parameters if present. And it will try to guess the size of the alignment zones by measuring certain key glyphs in the font:

- for the ascender zone, it will take the height of the lowercase f plus one unit,
- for the cap zone, the height of the uppercase O plus one unit,
- for the small cap zone, the height of 'o.sc' plus one unit,
- for the x-height zone, it will measure the lowercase o, and add one unit,
- the baseline zone defaults to -15,
- and for the descender zone, Glyphs takes the depth of the lowercase g plus one unit.

If Glyphs cannot find any glyphs to measure, the sizes for top zones default to 16, and to -16 for bottom zones. It is a good idea to double check if all your important glyphs reach into, but not beyond, the alignment zones.



Position	Size
719	9
699	9
489	9
0	-9
-209	-9

7.2.5 Custom Parameters

Click on the plus button to add custom parameters for the masters. You can also copy and paste parameters. Values are interpolated if defined in all masters. Master parameters override all font-wide settings and parameters,

but are themselves overridden by instance-level settings and parameters.

See section 17.3, ‘Custom Parameters’ (p. 180), for possible values and a detailed description of custom parameters.

7.3 INSTANCES

Instances are the actual fonts being created when you choose *File > Export* (Cmd-E). To add a new instance, click on the plus button in the lower left corner of the window, and pick an option from the menu that appears:

- *Add Instance* inserts a new instance called ‘Regular’ with the default values Weight 100, Width 100, and Custom 0.
- *Add Instance for Each Master* adds instances at the same design space coordinates as the masters set up in *File > Font Info > Masters*. The respective style names are then derived from the master names.

Select one or more instances and click the minus button to delete them. You can edit several instances at once if you Shift-click or Cmd-click the instance names in the list on the left. Rearrange instances by dragging them into a new position. The order has no significance.

For more information on interpolation, see section 12.3, ‘Setting up Instances’ (p. 143).

7.3.1 Is Active

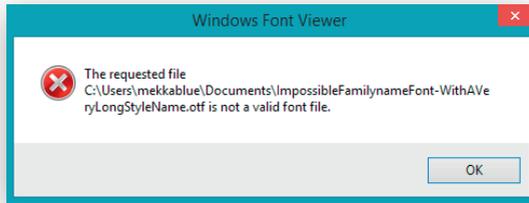
The *is active* setting controls whether this particular instance actually exports when you choose *File > Export* (Cmd-E). Deactivated instances will be displayed in gray in the sidebar, and are ignored at export.

7.3.2 Style Name

This is the style name as it will appear in the font menu of an application, e.g., ‘Bold Condensed’ or ‘Light Italic’. You can use a space in the style name, but non-ASCII characters may prevent the font from exporting. If you do need special characters in your style name, consider the *localizedStyleName* custom parameter. For more details, see its entry in the list of custom parameters in the Appendix (p. 189).

Also, keep the style name short. Some environments have difficulties with long font names. Microsoft Windows can declare a font invalid if the overall name (family and style name) exceeds a limit of 20 letters. In that case,

consider using a shortened style name and the parameters *preferredFamilyName* and *preferredSubfamilyName*. For more details, see their entry in the list of custom parameters in the Appendix (p. 193).



7.3.3 Weight and Width

With the *Weight* and *Width* drop-down lists, you set the values weight class and width class, displayed on the right next to them. These numbers can be used by an application to sort the fonts in a logical order, to facilitate the choice of fonts within a family, or for referencing font files in CSS files.



Some *Weight* menu settings yield the same weight class number. E.g., Thin, ExtraLight and UltraLight all cause the weight class to be set to 250. In cases like this, if you need to differentiate further, you can override the value calculation with the *weightClass* parameter. For further details, see its entry in the list of custom parameters in the Appendix (p. 202).

In Microsoft applications, values below 250 may lead to artificial boldening of the font. Many web browsers will only take the first digit of the weight class into account. Both weight and width class affect the order of fonts in the font menu in Adobe applications. Sorting occurs first by width, then by weight, e.g.:

- Condensed Light, Condensed Regular, Condensed Bold
- Light, Regular, Bold (normal width)
- Extended Light, Extended Regular, Extended Bold

7.3.4 Style Linking

Here, you can link the instance to the another instance of the same family (all fonts carrying the same family name), and define it as the other instance's Bold, Italic, or Bold Italic. An application may allow the user to switch between the linked fonts through the means of Bold and Italic buttons, keyboard shortcuts, or menu items.

To link one font to another, type the exact style name of the other font in the text field, and click one or both of the *Bold* and *Italic* check boxes.



Adobe applications will display all fonts of the same family (i.e., with the same family name), in the style menu, and will allow the user to switch between italic-linked fonts with Cmd-Shift-I, and bold-linked fonts with Cmd-Shift-B.

Mac applications such as TextEdit can switch to linked Bold and Italic styles via *Format > Font > Bold* (Cmd-B), or *Format > Font > Italic* (Cmd-I), respectively.

Microsoft Windows applications support only the basic four styles per font family, i.e., Regular, Italic, Bold, and Bold Italic. They are accessed via the 'B' and 'I' buttons in the toolbar of the Windows application, or the Ctrl-B and Ctrl-I keyboard shortcuts. If a font family is not style-linked at all, then each style will appear as an individual entry in the font menu. In that case, clicking the 'B' or 'I' buttons may create a synthetic Bold or Italic. Based on these conditions, we recommend the following style-linking strategy:

- The Bold, Italic, and Bold Italic styles of your font family should always be linked to the Regular. Within a given width class of a given font family, use the Bold linking only in this set of four fonts. E.g., 'Italic is the Italic of Regular, Bold is the Bold of Regular, Bold Italic is the Bold and Italic of Regular'.
- Other italic styles should always be linked to their upright (non-italic) counterparts, e.g., 'Medium Italic is the Italic of Medium'.

Some designers link the Semibold to the Light. If you choose to do so, only the Light will appear in the font menu of some applications, especially Microsoft Windows applications.

Therefore, users may be unaware that there is a Semibold, or that they can access the Semibold by selecting the Light in the menu and subsequently activating the Bold style. If you still do want to link Semibold to Light, enter the name of your Light style in the Style Linking text field of your Semibold instance settings, and check the Bold option next to it.

7.3.5 Interpolation

The *Weight*, *Width*, and *Custom* settings apply to the design space spanned between the Masters. For further details, see chapter 12, ‘Multiple Masters’ (p.141).

7.3.6 Custom Parameters

With instance-level parameters, you can produce different versions of the same font carrying different copyright notices, UPM values, family names, etc. Instance parameters override all master and font settings and parameters.

See section 17.3, ‘Custom Parameters’ (p.180), for a detailed description of custom parameters.

7.3.7 Instance Preview

At the bottom, you will see an interpolated preview of the string ‘Aang126’. If you want different glyphs in your preview, you can add an *Instance Preview* parameter with a list of glyph names that should be displayed instead.

7.4 FEATURES

7.4.1 OpenType Feature Code

OpenType feature code entered in *File > Font Info > Features* applies to the whole font. That is why all GPOS features that are calculated specifically for each instance, such as the ‘kern’ feature, are not displayed here.

You can add OpenType classes, OpenType features, and so-called prefixes to the code. Classes are named collections of whitespace-separated glyph names, features are the actual feature code in AFDKO syntax. The prefix is for all information that needs to stand outside an actual feature, e.g., lookup definitions you want to re-use further down in the code. The automatic feature generator will, by default, put the code for required language systems into the prefix.

7.4.2 Automatic Feature Code

Glyphs can automatically generate the code for many common OpenType features. Click *Update* in the bottom left window corner to initiate the automatic generation and ordering of OpenType features. The built-in feature generator relies on a naming convention and glyph name suffixes. E.g., all letters with names ending in ‘.sc’ will be put in the small caps features smcp and czsc. In most cases, adding the feature tag as glyph name suffix, will trigger the automatic generation of that feature, e.g., x.sso1 will trigger the automatic generation of the Stylistic Set 1 feature. Find a list of recognized suffixes in section 17.1, ‘Automatic Feature Generation’ (p. 175).

Some features and classes can be generated automatically, but need to be inserted manually. To do this, click on the plus button, and choose *All*, *AllLetters*, or *Capital Spacing* from the menu that pops up. The class *All* contains all glyphs, *AllLetters* contains all glyphs categorized as ‘Letter’. Adding the feature *Capital Spacing* (csp) adds extra space between capital letters in all-caps typesetting. It will also trigger the *Uppercase* class, containing all glyphs of category ‘Letter’, subcategory ‘Uppercase’.

The deprecated *Mathematical Greek* (mgrk) feature can be automated, but does not show up in the drop-down list of the plus button. In this case, you would add a feature by clicking on the plus button and choosing *Feature* from the drop-down list, rename the newly created entry ‘xxxx’ to ‘mgrk’, and activate its *Generate Feature Automatically* checkbox.

7.4.3 Manual Feature Code

To write features, prefixes, and classes manually, uncheck the *Generate Feature Automatically* checkbox, or create a new item by clicking the plus button at the bottom left. Edit the name of the feature, prefix, or class directly in the list on the left, insert feature code in the top right pane. For the feature code itself, Glyphs makes use of Adobe FDK syntax. You can test-run your features with the *Compile* button.

All features and classes will be added to the feature file with their respective names, as added in the left sidebar. The only exception is ‘kern’, which is added as a lookup inside the kern feature. See section 9.2.8, ‘Optional ‘kern’ Feature Lookup’ (p. 120), for more details.

The Adobe FDK feature file syntax is described in detail at: www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html.

While writing feature code, Glyphs will apply syntax highlighting. You can autocomplete AFDKO feature code keywords such as `substitute`, `ignore` or `lookup` by pressing the Esc or F5 key. If multiple completions are possible, a little menu will pop up under the word in question. Choose the completion with the up and down arrow keys, and confirm by pressing the Return key.

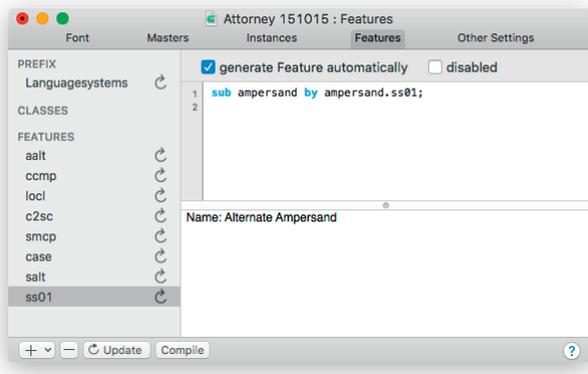
To remove features, prefixes or classes again, select one or more entries in the sidebar, and click the minus button at the bottom or press the Delete key. If you do not want the feature to export into the OpenType font, but still keep it, you can disable it. To do so, select a feature, prefix, or class, and click the *Disabled* checkbox at the top of the window.

The bottom right area is for notes. In Stylistic Set features (ss01 through ss20), you can use it to set the name of the Set. Type 'Name:', followed by the intended description. Alternatively, you can enter the complete featureNames lookup as described in the Adobe FDK syntax:

```
featureNames {  
    name <platformID> <scriptID> <languageID> "featurename";  
};
```

A featureNames block may contain several name entries, for different platforms, scripts, and languages. You can leave out languageID, scriptID, and platformID, in this order, if they are the same as the default platform 3 (Microsoft), script 1 (Latin), language 0x409 (American English). Keep in mind that as of this writing, Stylistic Set names are only supported by the applications making use of the Cocoa text engine in OSX 10.11 El Capitan or later.

Platform, Script and Language IDs are explained in Adobe's name table documentation: partners.adobe.com/public/developer/opentype/index_name.html



On a per-instance basis, specific OpenType features can be disabled with the *Remove Features* custom parameter. Or, OpenType features can be changed with the *Replace Feature* parameter. For more details, see the respective entry in the list of custom parameters in the Appendix (p. 195).

7.5 OTHER SETTINGS

7.5.1 Grid Spacing and Subdivision

Contrary to popular belief, decimal coordinates can be exported into PS-based OTFs.

The *Grid Spacing* value defines how coordinates get rounded. Standard is the value 1. When set to zero, no rounding will happen. This is useful if you want to keep very fine details, e.g., after applying the *Hatch Outlines* filter or in detailed dingbat fonts, or if you plan to scale your glyphs and want to minimize rounding errors. Higher values are helpful when creating a pixel font.

All tools and all modifications will consequently snap to the grid. Regardless of the grid settings, you can round all point coordinates of selected nodes or glyphs to integer numbers by choosing *Paths > Round Coordinates*.

The *Subdivision* value gives you the option to subdivide the Grid Spacing if you need finer steps but want the larger grid spacing as design orientation. The value indicates into how many compartments the main grid is subdivided, e.g., a *Grid Spacing* value of 100 and a *Subdivision* value of 5 will yield a subgrid with a 20 units step. Setting *Grid Spacing* to 1 and *Subdivision* to 10 will give your point coordinates one decimal.

7.5.2 Use Custom Naming

This option prevents the automatic replacement of glyph names with names as suggested in the built-in glyph database. This may be of importance where special workflow requirements apply. The option is set in all fonts imported from OTF files, and UFOs that are opened with Glyphs for the first time, if the *Keep glyph names from imported files* option is set in *Glyphs > Preferences > User Settings* (see 2.1.2, ‘User Settings’, p. 12).

Deactivating the *Custom Naming* option does not immediately activate the automatic replacement of names. To trigger the renaming, select all glyphs in Font view, and use *Glyph > Update Glyph Info*. Attention: This may invalidate imported or manually written feature code.

Only when the *Custom Naming* option is activated, can you set your own Unicode values. Otherwise, Glyphs will derive them from your glyph names. See section 6.3.6, ‘Unicode’ (p. 71) for more details.

7.5.3 Disable Automatic Alignment

This option disables the automatic alignment of components and the autosync of metrics for component-based diacritics. If you want to prevent accidental shifts of components in this case, you can lock them via the context menu. Newly imported files will have automatic alignment disabled if the corresponding option in the app preferences is set accordingly (see 2.1.2, ‘User Settings’, p. 12).

7.5.4 Keep Alternates Next to Base Glyph

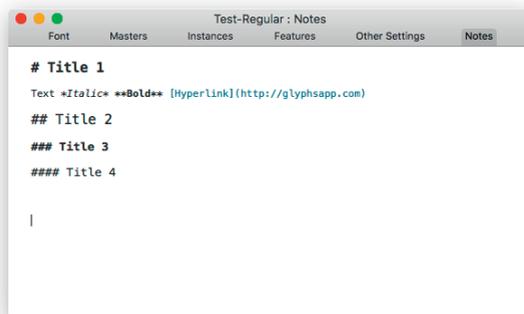
In Font view, glyph variations with name suffixes will stay next to the main glyph if this option is enabled. For example, h.ss16, h.alt, h.loclENG will be displayed right after h instead of being moved to the end of the category.

7.6 NOTES

7.6.1 Font Note

Text stored in the font note is only available inside the Glyphs file and does not export into the OpenType font. The content of the text field is equivalent to the value of the font custom parameter *note*, see its entry in the list of custom parameters in the Appendix (p. 190).

The text entry sports Markdown-aware highlighting for bold, italic, hyperlink, code and title formatting.



8 Reusing Shapes

8.1 COMPONENTS

Displayed as a gray glyph inside another glyph, components are an efficient way to re-use the shapes of glyphs inside multiple other glyphs. The main purpose of components is their use inside diacritic variations of the base letter, e.g., A in Ä, Å, Æ, Å, Å, Å, Å, and Å.

The original, usually outlined, glyph that the component points back to, is referred to as the 'base glyph' of that component. A glyph built with components is usually referred to as 'compound' or 'composite glyph'. Since the component always stays in sync with its base glyph, you only need to change the original outline, and the changes will be visible in all composites.

Components are converted into plain outlines before exported in OpenType/CFF fonts, since CFF does not properly support components. However, in TrueType-based fonts, components are kept, unless they overlap each other and the custom parameter *Keep Overlapping Components* is not set. For more details, see its entry in the list of custom parameters in the Appendix (p.188).

8.1.1 Building Compounds

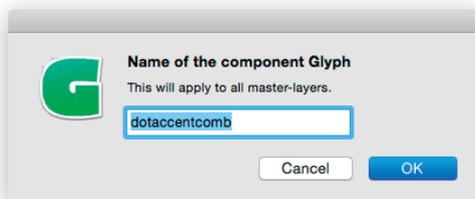
You can build compound glyphs by either creating a glyph with the proper name, e.g., 'acute', or by selecting existing glyphs and choosing *Glyph > Make Component Glyph* (Cmd-Opt-Shift-C) to rebuild the glyph as a composite with its default recipe. Glyphs will use a built-in database of glyph compositions to determine the components of the glyph in question. You can add components yourself to an existing glyph by choosing *Glyph > Add Component* (Cmd-Shift-C) and picking the base glyph in the subsequent dialog.

If you already have a letter that you want to turn into a component-based glyph, you can force the composition with *Glyph > Make Component Glyph*. The content of the letter will be deleted and rebuilt from scratch, then. This only works if the base glyphs of the individual components already exist in the font.

While multiple glyphs are selected, you can add the same component to all those glyphs at once, using the same menu command.

8.1.2 Turning Paths into Components

If the base glyphs are not already present in the font, you can create a new component from an existing path by selecting the path you want to turn into a component and choosing *Glyph > Component from Selection* or picking *Component from Selection* from the context menu. Glyphs will then suggest a name for the base glyph based on the decomposition information from the built-in glyph database:



Once you confirm the dialog, Glyphs will create a new glyph containing the selected path and place it as a component in the original glyph. In multiple-master setups, it will look for corresponding paths in all master layers, assuming that master layers have a compatible path order. This mechanism can be useful for deriving *dotaccentcomb* and *idotless* from *i*, for instance.

8.1.3 Recipes

When creating new compound glyphs, you can use composition recipes and force a non-standard composition. For this, you need to bring up the *Glyph > Generate Glyphs* dialog. There, you can build recipes in three ways:

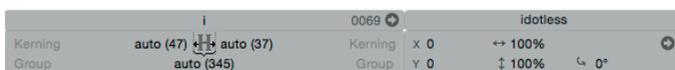
- *A=a* *base* → *composite copy*
- *x+dieresiscomb=xdieresis* *base + mark* → *mark compound*
- *s.sc+s.sc=germandbls.sc* *base + base* → *compound ligature*

In the first example, the ‘composite copy’, Glyphs will create a lowercase glyph called ‘a’ with an uppercase ‘A’ placed in it as a component. In the second example, the ‘mark compound’, a glyph called ‘xdieresis’ will be generated with the letter ‘x’ as the base, combined with the mark ‘dieresiscomb’. In the last example, the compound ligature recipe, Glyphs will create a glyph called ‘germandbls.sc’ with two letter components ‘s.sc’ next to each other.

8.1.4 Editing Components

Select a component by clicking on it. Press the Tab key to select the next, or Shift-Tab to select the previous component. Rectangular mouse selections ignore components, unless you simultaneously hold down the Option key.

The gray Info box will display component information: the name of the original glyph the component points to, its position, scale, and rotation. Open the original glyph by either double clicking the component or clicking on the arrow next to the glyph name. Change the component by clicking on its name. Glyphs will prompt you for the new glyph the component should point to.



You can edit component attributes by changing the values in the gray Info box, or by applying transformations from the Transformations Palette, such as aligning, scaling, rotating, and mirroring. Flipping mark components both horizontally and vertically will switch their top and bottom anchor association. I.e., a flipped top mark will attach to a bottom anchor and vice versa. This can, for example, be useful for reusing the bottom mark `commaccentcomb` in `gcommaaccent` (i.e., instead of using the default `commaabovetcomb`), where the comma accent is supposed to reside on top of the `g`.

You can move a selected component with the mouse or the arrow keys if it is not automatically aligned. Hold down the Shift key for increments of 10, and the Command key for increments of 100. Option-drag a component to duplicate it. Delete it by simply pressing the Delete or Backspace key.

8.1.5 Moving between Base Glyphs and Compounds

To edit the base glyph of a component, double click the component and Glyphs will place the original letter next to the compound glyph, and activate it for immediate editing. Alternatively, you can select the component inside the composite glyph and click the arrow symbol that appears in the gray Info box (Cmd-Shift-I).

Placeholders can indicate an 'empty base glyph', 'no base glyph', or a 'bad reference'. An empty base glyph is a base glyph without paths or components. Double clicking its placeholder

will open the empty glyph next to the current glyph, so you can edit it right away. 'No base glyph' means that the glyph the component is referring to does not exist in the font. Double click the placeholder to create the glyph, and insert it in the Edit tab, next to the current glyph. A 'bad reference' indicates a broken component reference, e.g., a circular reference, i.e., a component that points to the compound glyph in which it resides.



You can see in which compounds a glyph is used as a component by right-clicking or Ctrl-clicking on the glyph in Edit view to open its context menu, and choosing *Show all glyphs that use this glyph as a component*. All glyphs that contain a component pointing back to this glyph will be inserted next to it in the Edit tab. This command is also available in the context menu of placed components. All other compounds containing the same component will be inserted next to the current glyph.

8.1.6 Anchors

In mark compounds, accents are snapped into place if corresponding anchors are placed in both the base letter and the original mark. The letter needs an anchor (e.g., 'top'), and the mark needs a corresponding mark anchor, i.e., an anchor with the same name, but preceded by a leading underscore (e.g., '_top'). In the composite, the mark will then be positioned according to the relative anchor positions.

For many glyphs, default anchors are predefined in the built-in glyph database. They can be added to selected glyphs with *Glyph > Set Anchors* (Cmd-U). If you additionally hold down the Option key, the command changes to *Reset Anchors* (Opt-Cmd-U), then Glyphs will delete all present anchors and reset the default anchors. Setting and resetting anchors also works for multiple glyphs. For the automatic positioning of anchors, the italic angle of the master is respected.

You can add custom anchors by choosing *Add Anchor* from the context menu. You will immediately be prompted for a

Tip: *Edit > Select All* (Cmd-A) selects all paths, but ignores anchors and components. Selecting all twice in a row (hold down Cmd while you type A twice) also selects anchors and components.

name. Enter a name and confirm by pressing Return. You can change the name of an existing anchor either by selecting it and pressing Return, or by double clicking it. Then, you can start typing the new name:



Select an anchor in the base glyph to get a preview of the most common accents that may attach to this glyph. Similarly, select an anchor in the mark glyph and this accent is shown on all other glyphs in the same Edit view. If exactly one anchor is selected, press the Tab key to select the next anchor, or Shift-Tab to go to the previous anchor.

Change the position of an anchor by dragging it with the mouse, or selecting it and pressing the arrow keys. Add Shift or Command for increments of 10 or 100 units, respectively. Delete selected anchors by pressing the Delete key. Select an anchor and two points and choose *Paths > Align Selection* (Cmd-Shift-A) to horizontally center the anchor between the points. Again, Glyphs will respect the italic angle set in *File > Font Info > Masters* (Cmd-I).

Sometimes you need more than one ‘top’ anchor, e.g., in ligatures, or for fine-tuning positions of certain marks, or in circumflexcomb for different positions where marks can attach in Vietnamese double accents. Multiple anchors of the same kind need to be underscore-suffixed, e.g., ‘top_1’ and ‘top_2’, or something more descriptive such as ‘top_viet’ or ‘top_acute’. If you apply this naming scheme, you can choose a different anchor for a component by selecting it, clicking on the anchor symbol in the Info Box, and choosing an alternative anchor from the pop-up menu. The anchor symbol becomes visible if there is more than one anchor that fits the currently selected component:



To change the default anchors, or which marks are displayed in the mark cloud, see section 17.4.1, 'Global Glyph Data Changes' (p.204) in the Appendix.

8.1.7 Automatic Alignment

Letters built entirely from components, so-called compound or composite glyphs, can make use of automatic alignment, given that the option *File > Font Info > Other Settings > Disable Automatic Alignment* is off (see 7.5.3, 'Disable Automatic Alignment', p. 97). This means that both the placement of the components inside the compound, and the width of the compound are determined by the base glyphs and their anchors.

Single letters ('component copies'): If the base glyph is of the category 'Letter', a spacing combining mark, or a small figure (e.g., a denominator or scientific inferior), the resulting compound will be automatically aligned with the base letter, inheriting its positioning and sidebearings.

Letters with accents: If the first component points to a glyph containing an anchor whose name does not start with an underscore (e.g., 'top'), and the second component points to a glyph containing an anchor of the same name with a preceding underscore (e.g., '_top'), the resulting compound will be automatically aligned with the first glyph, the 'base glyph'. The glyphs of the subsequent components, the 'marks' will be automatically placed according to the positions of the corresponding anchors. This method of automatic alignment is also referred to as 'attachment'.

Multiple letters, with or without marks: The automatic placement and spacing of the letters following each other takes both the glyph widths and their kerning into account. For example, when building a onehalf fraction out of `one.numr`, `fraction`, and `two.dnom`, the fraction will look as if its parts were typed individually. To move them to a different position, you can either space and kern the individual glyphs, or you can disable the automatic alignment of a component via the context menu (see further below).

Or, you can use exit and entry anchors to create compounds. Then the beginning sidebearing is taken from the first base glyph, the trailing sidebearing from the last base glyph. The relative placement in between is then determined by the relative position of exit and entry anchors in the base

glyphs of the components. This way, you can place letters, but also letter parts precisely to each other.

Corresponding anchors will trigger automatic mark attachment feature code (mark and mkmk features). And for cursive connecting scripts such as Arabic, exit and entry anchors will trigger automatic code generation for the cursive positioning feature (curs). If you want to avoid automatic generation of such GPOS feature code, and use the corresponding anchors solely for compound alignment, you can prefix the anchor names with an arbitrary non-letter, e.g., #top and _#top, as well as #exit and #entry.

By their very nature, automatic alignment and metric keys are mutually exclusive, with one exception: You can add space to the sidebearing of an automatically aligned letter by using the ‘=+’ and ‘=-’ operators, followed by the amount of additional spacing. For more information, see section 9.1.3, ‘Metric Keys and Automatic Alignment’ (p. 116).

Some glyph categories are not automatically aligned, e.g., numerals. This, for instance, allows to re-use proportional figures for tabular figures or vice versa. You can force automatic alignment by Ctrl- or right-clicking a component, and choosing *Enable Automatic Alignment* from its context menu.

Also, automatic alignment of components is disabled as soon as there are any paths in the glyph. Once those paths are removed again, the components will snap back into automatic alignment. To explicitly disable automatic alignment of a component, Ctrl- or right-click on a component, and choose *Disable Automatic Alignment* from its context menu. You can disable automatic alignment for the base, but keep it for the subsequent marks. Thus, you can move the compound and change its width, while the marks still stay aligned to the base.

For a complete, font-wide deactivation of automatic alignment, go to *File > Font Info > Other Settings*, and check the *Disable Automatic Alignment* option.

8.1.8 Locking Components

For components that cannot or should not be automatically aligned, there is the option to lock them in their current position. To do that, select a component, and pick *Lock Component* from its context menu. Locked components

cannot be selected, preventing accidental shifts. To unlock a component again, right click or Ctrl-click it to bring up its context menu, and choose *Unlock Component*.

8.1.9 Decomposing

You can turn all components inside a glyph into editable paths by choosing *Glyph > Decompose Components* (Cmd-Shift-D). All components in the visible layer of the current glyph will be removed, and its paths and anchors will be inserted instead. Nested components will be decomposed as well.

Selecting *Decompose* from the context menu of a specific component only decomposes the selected component. In this case, nested components will not be decomposed.

8.1.10 Combining Paths and Components

As soon as there is a path in the layer, automatic alignment is disabled. Therefore, be careful when combining components and paths, because shifts may occur, especially if the base glyph of the component is changed. If you want to prevent shifts, consider rebuilding the glyph with components and anchors, thus enabling automatic alignment.

While dragging an object over a placed component, vector points of the component become highlighted. Nodes dragged above a component snap to the compound nodes. You can align a component with an outline node by selecting both and choosing *Paths > Align Selection*. While the outline node stays put, the component will be moved above the point so that the origin of the component will be aligned with the node. The origin is the point in the component glyph where the left sidebearing intersects with the baseline, or, if present, the position of an anchor named 'origin'. This can be useful for placing serif components.

8.1.11 Nesting Components

Glyphs allows you to nest components. For example, you can build the dieresiscomb glyph out of two dotaccentcomb components. Subsequently, you can use this compound dieresiscomb in higher-level compounds such as adieresis (ä).

Anchors will shine through in nested components, unless they are overridden by a specific anchor placement in a higher compound in the nesting chain. That means that you do not need to reset anchors in nesting components. E.g., if you are

building `oslashacute` from `oslash` and `acutecomb`, and `oslash` is composed of `o` and `slashlongcomb`, then the `acutecomb` can re-use the 'top' anchor of the `o`. You do not need to set another 'top' anchor in `oslash`, unless you do want a different mark placement.

8.1.12 Preferred Marks for Glyph Composition

When automatically building compounds, Glyphs will prefer marks that carry the same name suffix as the compound. For instance, when composing `adieresis.sc`, Glyphs will prefer `dieresiscomb.sc` to `dieresiscomb`, if it is available in the font.

For building compound uppercase letters, marks with a special 'case' suffix are preferred. E.g., Glyphs will prefer `dieresiscomb.case` over `dieresiscomb` for `Odieresis` (Ö).

When building combinations for `i` and `j`, make sure both `idotless` and `jdotless` are in your font. This is because the dot is typically not included when `i` and `j` receive accent marks. For `i` and `j` diacritics, Glyphs will prefer marks carrying a 'i' or 'narrow' suffix. The presence of `i`, `j`, `idotless`, `jdotless`, and combining marks will also trigger the creation of the `ccmp` feature in *File > Font Info > Features* (Cmd-I).

8.1.13 Underscore Components

If you are adding glyphs to your font that are not supposed to appear in the compiled font, but are solely used as parts in other letters, then it is advisable to prefix their names with an underscore character, e.g., `_leftBottomSerif` or `_A.ogonek`. This simplifies finding such glyphs through, for instance, a Smart Filter (see 6.5.4, 'Smart Filters', p.76), or the Search function (see 6.1.3, 'Searching for Glyphs', p.66). Also, when glyphs are generated with a preceding underscore in their name, they will be created with the *Export* option off.

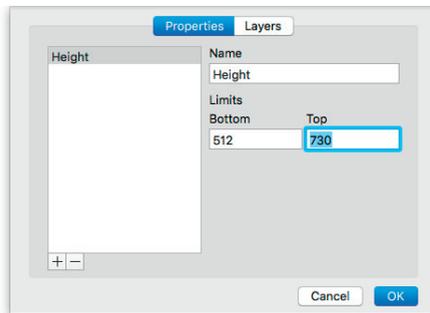
8.2 SMART COMPONENTS

Smart components were developed with Asian scripts in mind, but can be used for any shape that is reused often with modifications. In CJK ideographs or scripts like Tibetan, letters sometimes contain adapted smaller variants of other letters. In order to take care of this adaptation that makes it fit into the larger glyph, the smart component technology allows you to set up a smart glyph with local masters, i.e., masters for this letter only.

8.2.1 Setting up Smart Glyphs

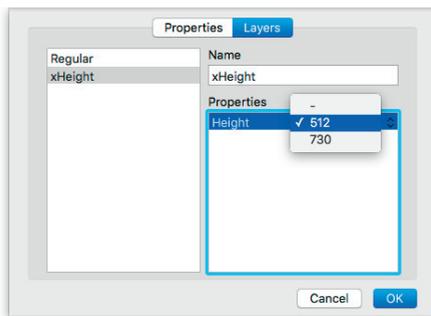
A smart glyph is a glyph that can be set up with local masters, i.e., non-master layers that function as interpolation extremes. A smart glyph can subsequently be interpolated between those layers when inserted as a smart component in another glyph. Any CJK radical and any Korean base glyph will count as a smart glyph by default. Apart from that, any glyph that starts with ‘_part’ and carrying a dot suffix, e.g., _part.001 or _part.arch, will be treated as a smart component.

In a smart glyph, add several layers with shape variations, name them in a way that makes sense to you, e.g., ‘Wide’, ‘Deep’, ‘Narrow’, etc. Then, choose *Show Smart Glyph Settings* from the contextual menu (Ctrl-click or right-click) or press Cmd-Opt-I to open a dialog sheet with options for the current smart glyph. In the *Properties* tab, you can specify names and ranges of local interpolation axes, i.e., interpolations for this glyph only. We advise to give each property a sensible name, and also a *Bottom* and a *Top* value that makes sense to you. For example, if you are setting up a part with an adaptable descender depth, you may want -100 as *Bottom* value, and 0 as the *Top* value. Or a letter part that can adapt its width, e.g., the shoulder in n and m, you could take a *Bottom* value of 0 for the minimum width, and a *Top* value of 100 for the maximum width. We recommend using measured values wherever possible, e.g., the actual height of an ascender for its Height property:



Once you have set up properties, you can then switch to the *Layers* tab and assign an extreme for every property to each layer. E.g., the Regular layer is set as bottom width and top depth, the Wide layer as top width and top depth, and the

Deep layer as bottom depth and bottom width. Make sure all properties in all layers are set to an extreme. When used as smart component in a different glyph, the shapes can be interpolated between the layers you assigned in the *Layers* tab, and along the axes you specified in the *Properties* tab.



While each smart glyph master needs to be a separate layer, you do not need to draw every extreme in a multiple dimension setup. E.g., if you have two properties like Height and Width, each of them interpolating between 0 and 100, you only need layers for three of the extremes. For instance, if you have 0 Height / 0 Width, 100 Height / 100 Width, and 0 Height / 100 Width, you do not need to draw a layer for 100 Height / 0 Width anymore, unless you are not content with the resulting extrapolation.

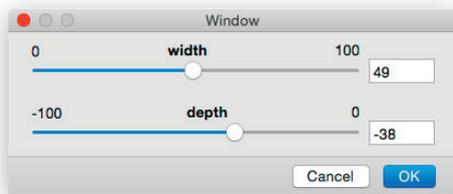
Glyphs recognizes two special properties, 'Width' and 'Height', spelled with a capital letter. Using these names enables you to use bounding box scaling of placed smart components. To use the bounding box, activate *View > Show Bounding Box* (Cmd-Opt-Shift-B).

8.2.2 Adding Smart Components

Inside a regular glyph, you add smart components by adding components as described in section 8.1.1, 'Building Compounds' (p.98), only this time, you pick smart glyphs, i.e., CJK radicals or glyphs that start with '_part'. This is easily achieved by simply typing an underscore in the search field of the *Glyph > Add Component* dialog (Cmd-Shift-C). It will then list all glyphs that start with an underscore. It should be easy to pick out the desired _part glyph then.

8.2.3 Smart Component Settings

In a glyph containing smart components, you can Ctrl-click or right-click a smart component, and choose *Show Smart Component Settings* from the context menu, or press Cmd-Opt-I. This brings up a dialog that allows you to interpolate the smart component in place. Sliders and number fields control the property values specified in the individual smart glyphs. Any changes are visible immediately.



You can also extrapolate if you type numbers beyond the top and bottom values in the number fields. You can also step through numbers with the up and down arrows. Hold down the Shift key for increments of ten.

8.2.4 Part Anchors

When multiple pieces are combined in one compound glyph, their respective positions can be automatically aligned if you use anchors with corresponding names, i.e., an anchor with a name starting with a letter in the base smart glyph, e.g., 'connect', and an anchor with the same name except for a preceding underscore in the connecting smart glyph, e.g., '_connect'. You can use both kinds of anchors in the same smart glyph in order to chain smart components inside a glyph.

8.3 CORNER AND CAP COMPONENTS

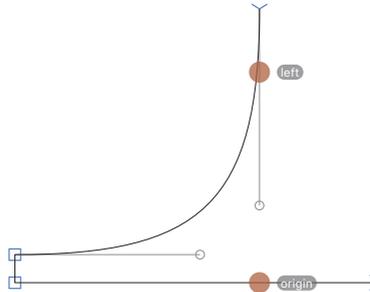
8.3.1 Corner Components

Corner components are open path fragments that can be dynamically fitted into an outline. The main purpose for corner components is to facilitate the construction and management of serifs.

In order to set up a corner component, you need to create a glyph with a name that starts with '_corner', followed by an arbitrary dot suffix, e.g., '_corner.leftSerif'. Inside the glyph,

Tip: employing both left and right anchors in the corner component allows building of cupped serifs ready for automatic flex hinting.

you draw an outline around the origin point, or, alternatively, around an anchor named 'origin'. Optional 'left' and 'right' anchors control the amount of distortion for adjusting the path fragment to fit into the receiving path:



The path direction, as indicated by the arrowheads at the open path ends, should match the orientation of the intended target paths. That means that if you are constructing a lower left serif, your serif path should start at the top point, bend around the origin, and end in the lower right point. To change the path direction, select the path, and choose *Reverse Selected Contours* from the context menu.

Once you are done with the corner component glyph, you can then proceed to insert the corner component into the outline of another glyph. To achieve this, select exactly one corner node in the outline of the glyph, and choose *Add Corner* from its context menu. The path fragment will immediately be bent into the corner, and also adapted to the slant of its surrounding outline segments.



You can press the Delete or Backspace key to remove it, you can copy the corner component into your clipboard and paste it on another node, or you can change its settings in the gray Info box. Via the Info box, you can switch to the original corner component glyph by clicking on the little arrow symbol. You can change which corner component is used by clicking on the name, as well as the scale, and the alignment of the corner component. The alignment controls which side of the corner component stays put while the other side gets

bent into the outline of the target path. You can have a corner component:

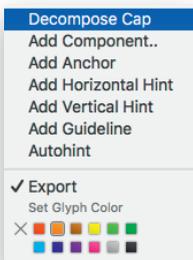
-  left-aligned, i.e., aligned with the stroke the entrance point is put upon, and the exit stroke stays put, which is typically what you want for a lower left or top right serif;
-  right-aligned, i.e., aligned with the stroke the exit point is put upon, and the entry stroke stays put, which is usually what you want for top left or bottom right serifs;
-  center-aligned, i.e., rotated between both alignments, so neither entry nor exit stroke stays put, which makes sense for ink traps and the like.

Generally, the presence of 'left', 'right' and 'origin' anchors, the open path direction, and the alignment setting determine the rotation and positioning of the corner component on the target path. For simple serifs, you can influence the adjustment and positioning of the curve in slanted stems, for instance, when you need to control how far a serif can stick out horizontally at the end of a slanted stem. To do that, add an anchor called 'left' or 'right' precisely above the origin. Pick 'left' for a left-aligned corner component, 'right' for a right-aligned one, or both for a center-aligned corner component or cupped serif. The distance between the origin and the anchor controls the amount of adjustment: Moving it up will shorten the corner components on an acute position, and elongate it on a blunt one. Moving it down will achieve the opposite. Cornered slab serifs (as opposed to bracketed serifs) will need these anchors exactly on the joint between vertical and horizontal segments.

Tip: the quickest way to mirror a corner component is to select it and press the mirror buttons in the Transformations palette.

If your design permits it, you can mirror corner components and reuse them on the opposite side. I.e., you can reuse a left serif on a right corner if you set its horizontal scale to -100%.

To decompose a path with corner components into its calculated state, you can remove overlaps by choosing *Filter > Remove Overlap* (Shift-Cmd-O). Decompose individual corner components by selecting them, and choosing *Decompose Cap* from the context menu.

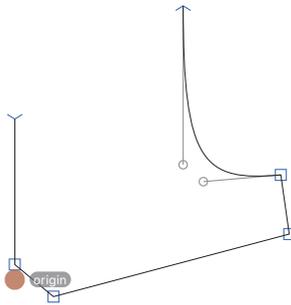


8.3.2 Caps

Caps work very much like corners except that they connect to two adjacent nodes rather than a single one. The idea behind

Caps is that they can be used for spurs, finials, or head serifs like on the top of a lowercase u.

In order to create a Cap, generate a glyph with a name that starts with an underscore, followed by 'cap' and an arbitrary dot suffix, e.g., `_cap.headSerif`. Draw an open path around the origin point, with the connecting ends coming out on or above the baseline, i.e., a head serif would need to be drawn flipped (turned 180°) and on the baseline. Put the entry point of the open path on or above the origin. If necessary, change the path direction by selecting the open path and choosing *Reverse Selected Contours* from its context menu. Again, the origin can be the origin point of the glyph, i.e., where the baseline intersects with the left sidebearing, or it can be an anchor called 'origin'.



In a target glyph, select exactly two adjacent nodes, e.g., the top nodes of the rectangle representing a stem. From the context menu, you can now choose *Add Cap*. The application will present a list of all available '_cap' glyphs. Pick the one you want and click the *Select* button or press the Return key to confirm the dialog. The cap component will get bent into the receiving shape according to its position relative to the baseline. Caps work best if the two receiving nodes have the same horizontal distance as the two end nodes of the open path inside the cap.

You can control the way the cap is adapted to the target path by clicking on the gray selection knob, and choosing options in the gray Info box. You can change the scale by entering a new percentage value for the vertical or horizontal scale. The *Fit* option scales the cap horizontally so that the open ends fit on to the two nodes of the target path. The option will disable the horizontal scale value. Change

alignment by clicking on the alignment symbols. Aligning to the left or right will orient the Cap to one side of the receiving path and take positions of the 'left' and 'right' anchors into account. The position of both 'left' and 'right' anchors, vertically relative to the baseline and horizontally relative to the start and end points, determines how the cap is bent onto the receiving nodes of the target path.



To change the cap, click on the name in the Info box, and choose a different cap in the following dialog. To open the cap for editing, click on the little arrow symbol.

To decompose a cap and insert it into the host path, select it and choose *Decompose Cap* from its context menu. To do the same with all corner components present on the current layer of a glyph at once, choose *Filter > Remove Overlap* (Cmd-Shift-O). To remove a cap from an outline, select it by clicking on its gray knob, and press Delete or Backspace.

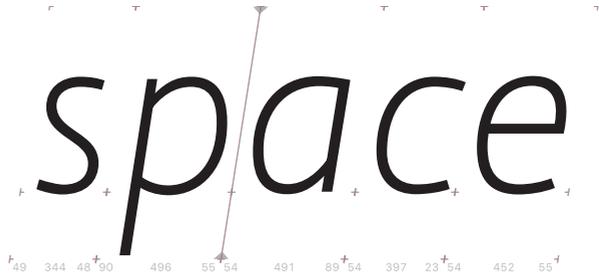
9 Spacing and Kerning

9.1 SPACING

Spacing is the process of adjusting the sidebearings of each letter to achieve an even rhythm in a line of text. There are no fixed rules for adjusting the white space. However, unless you are working on a monospaced font, similar or same shapes should have the same sidebearing, e.g., the D will usually have the same LSB as the H, but its RSB closer to or the same as in the uppercase O.

'LSB' stands for the left, 'RSB' for the right sidebearing of a glyph.

In Edit view, you can trigger the display of spacing information by activating *View > Show Metrics* (Cmd-Shift-M). When a glyph is active, this option shows the rectangle that indicates the width of the glyph and the vertical measurements of the master.



9.1.1 Spacing Shortcuts

There are keyboard shortcuts to change the spacing of the current letter in Text mode (shortcut T). Hold down the Ctrl key and use the left and right arrow keys to change the LSB. Cmd and arrow keys change the RSB. Hold down both Ctrl and Cmd keys to change both LSB and RSB simultaneously, effectively moving the glyph inside its width. Add the Shift key to manipulate in increments of 10 units.

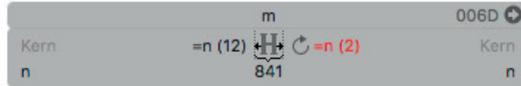
Easy to memorize: the Ctrl key is located on the left, the Cmd key on the right. The keys are associated with the left and right sidebearing, respectively.

The shortcuts for changing the LSB collide with default shortcuts for switching between OSX Spaces. You can change or deactivate the Spaces shortcuts in the System Preferences.

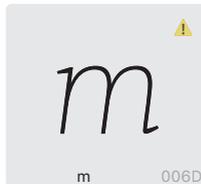
9.1.2 Metric Keys

In order to link metrics between glyphs, you can use so-called metric keys instead of mere numeric metric values. Put the name of the glyph you want to link to in the respective sidebearing or width field and it will automatically adopt the

sidebearing or width of the linked glyph. Note that keyed metrics are not kept in sync automatically. You need to select the glyphs that need updates and select *Glyph > Update Metrics* (Ctrl-Command-M) or *Glyph > Update Metrics for all Layers* (Ctrl-Opt-Command-M). In Edit view, out-of-sync metrics are colored red in the Info box and show an update button. Clicking on the update button updates only the metric key next to it.



In Font view, glyphs which are not in sync are marked with a yellow warning sign in the top right corner of the cell:



You can even enter simple calculations into the sidebearing field. Calculations need to start with an equals sign (=). For instance, '=n+10' will take the same sidebearing of n and add 10 units, '=n-10' subtracts 10 units, '=g/2' yields half the sidebearing of g, and '=v*2' doubles the sidebearing of v. Simply '=n' has the same effect as writing 'n' into the field. And equating with a number, e.g., '=20', allows to update to a predefined width or sidebearing. This can be useful for a monospaced font where the width of all glyphs must be the same, or for a connecting script font where the sidebearings must stay at a fixed value.

Use a double equals sign (==) instead of a single equals notation to specify a local metrics key. This means that the specified calculation is valid for the respective master only. This is necessary if you want different keys in different masters.

Use the pipe character (|, Shift-backslash on a U.S. keyboard, Opt-7 on a German keyboard, Opt-1 on a Spanish keyboard) to reference the opposite sidebearing. E.g., in the LSB of u, you can enter '=|n' to use the RSB of n for the LSB of u.

9.1.3 Metric Keys and Automatic Alignment

In order to additionally manipulate a sidebearing that is already governed by automatic alignment, use the equals sign followed by the respective mathematical operator and the number indicating the additional amount of whitespace, e.g., ‘=+20’ or ‘=-10’. This structure adds to, or subtracts from, the value that is calculated by automatic alignment.

This can be useful in cases where you need to extend a sidebearing to accommodate specific diacritic marks, but want to keep the compound auto-aligned. Typical examples in Latin typography are glyphs such as lcaron or dcaron, which employ a vertical caroncomb.alt on the right, and may therefore need additional sidebearing on the right.

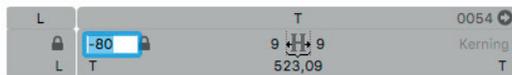


9.2 KERNING

If the spacing has been done properly, most glyphs fit well next to each other. But some glyph pairs still need specific adjustments. Usually, glyphs with a lot of white space are problematic. E.g., V before A, L before W, or T in combination with an unaccented lowercase letter like o will look like there is too much space between them. This is where kerning comes into play. Kerning is the adjustment (increasing or decreasing) of distance between two specific letters.

9.2.1 Ways to Kern

To define these kerning pairs, switch to the Text tool (shortcut T), type both glyphs in the Edit view and place the cursor between them. The left field (labeled *Kerning*) in the Info box will show the kerning value for the combination of the previous glyph with the active glyphs. On the opposite end of the Info box, the *Kerning* field shows the kerning value for the combination of the active glyph with the following one. Just click and type a value there.



Tip: Again, Ctrl (to the left of the Option key) corresponds to the left side of the glyph, Cmd on the right to the right side.

Or use the keyboard shortcuts for much greater convenience. Ctrl-Opt-arrow keys changes the kerning towards the letter on the left side of the current letter, while Opt-Cmd-arrow keys

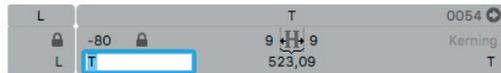
change the kerning with the letter following on the right. Add the Shift key for increments of 10.

9.2.2 Kerning Groups

Many glyphs look similar and need the same kerning values. Kerning groups capture these similarities and help you reduce the number of pairs that need to be set manually. Kerning then applies not just to pairs of glyphs, but of groups of glyphs. This way, you can not only kern A and T with each other, but all accented A and T glyphs as well, given that they all reside in the same groups.

Tip: The ‘Set Kerning Groups’ script from github.com/schriftgestalt/Glyphs-Scripts can create kerning groups for you.

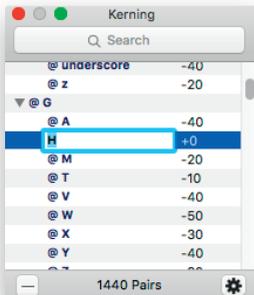
In Glyphs, kerning classes are not edited as glyph lists. Instead, the class membership is defined as a glyph property. For example, put O in the left kerning group field for all glyphs that look like an O on the left (like C, Ccedilla, G, Odieresis, and O itself). Add a value to each glyph, even if it remains the only glyph in its group.



The small extra panel on the left of the Info box displays kerning info for the glyph to the left of the cursor: It shows the name of the left glyph, its group lock for the current pair, and its kerning group at the bottom.

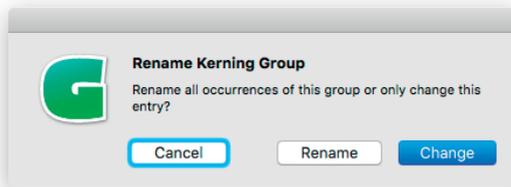
You can introduce exceptions to group kerning by opening the group locks in the gray Info box. Thus you can have different kerning pairs for ‘To’ and ‘Tö’, for example. Close a group lock to remove the kerning exception for that glyph. Within one pair, the kerning can be an exception for one glyph while it applies to the whole group for the other glyph. E.g. the kerning value defined for ‘Tö’ can be an exception for ö, since it does not apply to the other o glyphs. Thus, for ö, the (left) lock is open. However, the value does apply to all T’s including variations like \ddot{T} or Tj . So, for T, the lock remains closed.

You can rename a group and keep all its kerning values in the new group name by renaming the group in any of its entries in *Window > Kerning* (Cmd-Opt-K). To rename a group, double click, or select and tab into, the name in the window, and type a new name. Glyphs will then ask you to confirm the renaming of the group. E.g., if you rename the left @H group



to @B, you will turn all group kernings with @H on their right side (such as @/slash-@H) into group kernings with @B on their right side (e.g., @/slash-@B).

Alternatively, and in the same dialog, you can choose to only change the kern pair in question, by confirming the change of the pair rather than its renaming. In that case, the kerning groups remain as they are, only the kern pair in question is moved to a new group. E.g., you have accidentally kerned @d-@h to fit the dcaron, and now there is too much space between two following ascenders. To resolve this, you can assign @dcaron as a new right group to dcaron, and then rename the @d-@h kerning pair to @dcaron-@h. In the ensuing dialog, confirm by clicking the *Change* button.



9.2.3 Finding and Viewing Kerning Pairs

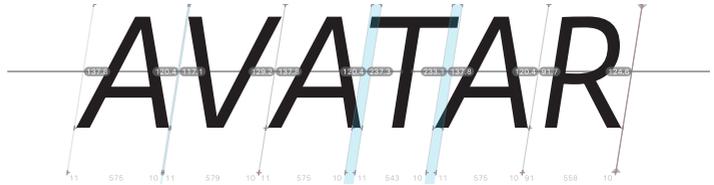
The Kerning window (*Window > Kerning*) lists all available kerning pairs, sorted by the first glyph or glyph class of the pairs. Groups are marked with an at sign, e.g. '@A' for the A kerning group, and displayed in dark blue. Individual glyphs are displayed in a softer beige.

Clicking on a pair in the Kerning window displays it in the current Edit tab, provided that either the Kerning **To*** or Locked Kerning **To^l** in the bottom right corner of the window is active. If the currently selected pair contains a group, you can choose *Show All Glyphs* from the gear menu to insert all possible glyph combinations for the current groups in the front-most Edit tab.

The search field on top of the Kerning window allows you to narrow down the displayed pairs. Clicking on the magnifying glass symbol gives you additional options, such as searching only for one side, only for groups or glyphs, etc.

You can color mark the kerning between glyphs in a sample text in Edit view if you display the measurement line via *View > Show Measurement Line*. For more details, refer to section 3.9, 'Measuring' (p. 38). If the *View > Show*

Metrics (Cmd-Shift-M) option is on, and the zoom level is large enough, small color markers at the baseline will be displayed.

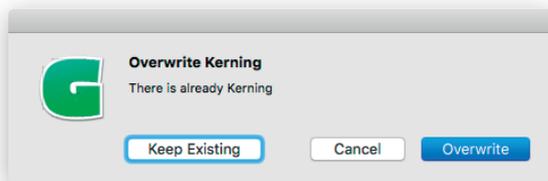


9.2.4 Deleting Kerning Pairs

In the Kerning window, clicking on the minus button in the bottom left corner deletes all selected kerning pairs. For the current glyph pair (i.e., the glyphs to the left and right of the cursor), you can simply delete the kerning value in the gray Info box (Cmd-Shift-I).

9.2.5 Copying Kerning Pairs

To copy kerning pairs from one font master to another, first select the kerning pairs you want to copy in *Window > Kerning* (Opt-Cmd-K). To select all kerning pairs, click in the Kerning window to put the focus on it, and choose *Edit > Select All* (Cmd-A). Then, choose *Edit > Copy* (Cmd-C), switch to the receiving font master, and paste (Cmd-V) the values into the Kerning window. A dialog will ask you what to do if existing kerning pairs are going to be overwritten.



9.2.6 Cleaning up Kerning

After removing glyphs or after importing, invalid or impossible kerning pairs may be left over. You can remove them with the *Clean Up* function from the gear button in the Kerning window.

9.2.7 Compressing Kerning

If you have kerned a few glyphs with each other before having set kerning groups, you can convert the singleton kerning pairs into group kerning with the *Compress* function of the gear button in the Kerning window. The function will delete the kerning between individual glyphs, and recreate it with their respective kerning groups wherever possible. Compressing kerning will keep explicit kerning exceptions that differ from the group kerning, but will remove exceptions that have the same value as corresponding group kernings. You may need to repeat the compressing process in order to catch all singletons.

E.g., assuming you have a kern pair \tilde{T} -m, i.e., kerning between an uppercase Tcaron (right group: @T) and a lowercase m (left group: @n), compressing it once will turn it into a kern pair of the @T group and the lowercase m, compressing it a second time will turn it into @T kerned with @n. Now the same kerning will work with all glyphs in the @T right group and the @n left group. However, it will keep exceptions such as uppercase T kerned with lowercase ñ, provided it has a different value than \tilde{T} -m.

9.2.8 Optional 'kern' Feature Lookup

In *File > Font Info > Features*, you can add a feature called 'kern'. Whatever you enter here will be added as a separate lookup called 'kernCustom' at the end of the kern feature in the features file. Since it is font-wide feature code, this additional kerning is not interpolated, but simply added to existing, interpolated kerning pairs. This is a good place for contextual kerning, which sometimes is needed for adjusting punctuation between certain letters, or situations where negative kerning would otherwise eat up the space, e.g.:

```
pos f' 60 space [T V W];  
pos L' -50 quoteright' 60 A;
```

In the first example, the f-space pair receives an extra 60 units only before T, V, W. In the second line, the combination LA is treated as follows: the quoteright is pushed into the L by 50 units, and the A is moved to the right relative to quoteright by 60 units. Effectively, the quoteright is moved to the left and 10 units extra space are added between L and A. This only happens in this exact constellation. I.e., the second line does not affect the combinations L'O or l'A.

Tip: You can reuse the kerning groups by prefixing the group name of the left glyph with @MMK_L_ and the group name of the right glyph with @MMK_R_. E.g., the L'A example here could read:
pos @MMK_L_L' -50 quoteright' 60 @MMK_R_A;
This assumes the right group of L is called L and the left group of A is called A.

10 PostScript Hinting

10.1 HINTING

PostScript hinting is a method to improve display at low resolutions for fonts with PostScript outlines. These will usually be OpenType/CFF fonts with an .otf file name suffix.

The eventual picture on the screen is created by software called the rasterizer. In PostScript-based fonts, the rasterizer needs to be relatively ‘smart’, because the fonts are said to be rather ‘dumb’. But we can place ‘hints’ in the font, which the rasterizer can use to improve the rendering.

Most hinting information revolves around determining which part of a letter is a necessary stroke element and should not be omitted in small sizes. In order to achieve this, there are two kinds of hints. Firstly, general information that applies to the entire font. This is referred to as ‘font-level hints’ or ‘font-wide hints’, and encompasses standard stems and alignment zones. Secondly, there are little pieces of information placed inside a glyph that help the rasterizer stretch the outline across the pixel grid. These are called ‘glyph-level hints’, and can be either stem hints or ghost hints.

Best practice is to choose good font-level hints, and subsequently let an algorithm called the ‘autohinter’ find the glyph-level hints for you.

Hinting only makes sense if the font has repeated regular features. If the font is very irregular, like many handwritten fonts are, or like ornamental and grunge fonts, then hinting cannot help improving the rendering. Also, if your font is intended for exclusive use in environments where hinting information is ignored, like displays with a very high resolution, or on Apple hardware running OS X or iOS, then hinting will only make the font file larger. In cases like these, it is advisable to *not* hint the font.

Keep in mind that the intention of PostScript hinting is to create a sharper, more consistent pixel image at low resolutions. That means that the outline will be distorted to achieve a better fitting on the pixel grid. In other words, hinting *does not preserve shapes*, on the contrary. If you have a font where the preservation of the shape is more important than a crisp pixel image, such as in connecting script typefaces and in icon fonts, hinting does not make sense.

10.2 FONT-WIDE HINTS

Before you do glyph-level hinting, you need to define a set of parameters that apply to all hinting throughout the font. These font-level hints are stored in the so-called ‘PostScript Private Dictionary’ inside the exported font. For an in-depth discussion, see the following links:

- partners.adobe.com/public/developer/en/font/T1_SPEC.PDF (specifically pages 35 – 45),
- vimeo.com/38364880 (video of a presentation about PS hinting by Miguel Sousa from Adobe, approx. 35 min).

10.2.1 Standard Stems

Stem widths are the thicknesses of your letter strokes.

A *vertical* stem is the width of a vertical stroke of a letter, e.g., the thickness of the uppercase I, or the thicknesses of left and right curves of an O. A *horizontal* stem is the thickness of a horizontal stroke movement, e.g., the serifs or crossbars of uppercase A and H, or lowercase t and f, or the upper and lower curves of an uppercase O.

And finally, *standard* stems are average values, as representative as possible for as many stem widths in the font as possible. The autohinter needs good standard stem values in order to recognize the stems and insert glyph-level hints automatically. And the screen rasterizer can make use of these values to optimize the pixel rendering, especially synchronizing stem thicknesses across the whole font at low resolutions.

Try to find as few as possible, and as representative as possible values for your horizontal and vertical stem widths and enter them in *Edit > Font Info > Masters* (Cmd-I), in the *Vertical Stems* and *Horizontal Stems* fields. If two values are close to each other, it is a good idea to merge them into one average value. You can quickly measure the thickness if you select two nodes and take a look at the gray Info box (Cmd-Shift-I) or by switching to the Measurement tool (see 3.9, ‘Measuring’, p. 38).

For instance, if you measure 68, 71, 72, 74, 75, 82, 83, and 85 for your vertical stems, then you would pick 75 or 80 as standard vertical stem, because either would be a good median value for most of the stem measures. By using a single stem value, the stems will scale more uniformly across low PPMs.

PPM stands for pixel per em, and is a measurement for the screen size. It effectively tells you the vertical size of your em in pixels.

Theoretically, up to twelve stem width values can be considered for each orientation. But the best practice of trying to find as few as possible will typically either result in a single representative value for all stems, or in two values: one for lowercase and one for uppercase letters, or (in the case of horizontal stems) one for an average horizontal stroke, and one for the serifs. Use a second or third value only if it is acceptable that the associated stems will have different thicknesses at the same pixel size. For instance, if you have a vertical standard stem set at 70, and another one at 80, the first stem may be displayed as two pixels wide, while the other stem may get three pixels at a certain pixel size.

The first value you enter in the *Horizontal Stems* or *Vertical Stems* is the most important one. Use a value that represents your most-used glyphs, typically the lowercase letters. This value is also used by other functions in the application, such as the Cursify algorithm or the Rounded Font filter. Any values that follow are exclusively used for hinting, the horizontal stems can also play a role in TrueType hinting. See section 11.3, 'Manual Instructions' (p. 133), for more details.

In a Multiple Master setup, values in individual masters are interpolated, so make sure you enter the values in the same order in each master.

10.2.2 Alignment Zones

When your font is rendered with very few pixels on a computer screen, all the x-heights should align to the same height, i.e., use the same amount of pixels vertically. The same applies to ascenders of letters like f, h, or k, and to descenders of g, p or y, and to the heights of all capital letters. And of course, all letters should share the same baseline when rasterized at a low resolution.

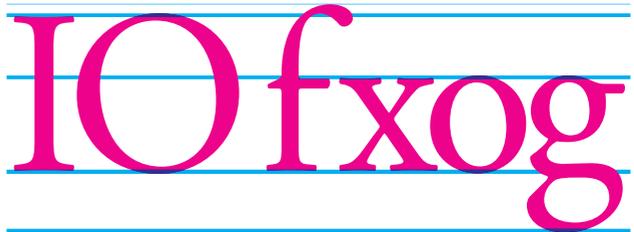
But all these letters usually do not really align precisely. For instance, the bottom of a lowercase o will extend slightly below the baseline, while the serifs of an n may sit exactly on it. Or the apex of an uppercase A may extend a little bit beyond the height of an uppercase H. This difference, usually some ten to fifteen units, is commonly referred to as 'overshoot'.

Alignment zones are a way to tell the rasterizer about the overshoots. At small pixel sizes, we do not need to see overshoots, so their display should be suppressed. Or, more

precisely, at low resolutions, any path constellation (a) with a horizontal stem or ghost hint attached to it, that (b) reaches into an alignment zone will be vertically aligned to the zone position.

Alignment zones take two values: a position and a size. The position is the vertical height of the zone, usually the vertical metrics, like x-height or ascender. The position is sometimes also referred to as the 'flat edge' of a zone. The size is the thickness of the maximum overshoot that may appear at that position. If the overshoot extends *above* the position (e.g., x-height, small cap height, cap height, ascender), the size value must be positive. Such zones are referred to as 'top zones'. If, however, the overshoots extend *below* the position (e.g., baseline, descender), the size must be negative and we call them 'bottom zones'.

A typical alignment zone setup: top zones with positive widths at ascender, cap height and x-height; bottom zones with negative widths at baseline and descender.



Alignment zones should be as small as possible, so do not try to make them larger 'to be on the safe side'. In any event, a zone must not be larger than 25 units. You can have a maximum of 5 top zones and 6 bottom zones. Zones must not overlap. There must be a minimum distance of one unit between them, the larger the better. The baseline zone must have a position value of zero.

If you make use of an alternative grid, i.e., if you employ Grid Spacing or Subdivision values other than 1 (see 7.5.1, 'Grid Spacing and Subdivision', p.96), then you need to extend the scope of your zones by one unit in both directions in order to catch potential small rounding errors for vertical node positions. I.e., the position must be shifted by one unit, and the size by two units. Only the baseline zone must be kept at position zero, while its size is increased by one unit.

More precisely, the maximum size of an alignment zone is constrained by the blueScale value (see below), which implies that no zone must be larger than $240 \div (240 \times \text{blueScale} - 0.98)$.

10.2.3 Custom Parameters

Apart from the alignment zones and standard stems, there are more optional parameters in the Private Dictionary: *blueScale*, *blueShift*, and Family Alignment Zones. In Glyphs, you can set these values as custom parameters. For a detailed discussion of what they do and how to apply them, please see their respective entries in section 17.3, ‘Custom Parameters’ (p. 180).

10.3 AUTOHINTING

If the font-wide parameters, i.e., alignment zones and standard stems, are set properly, you can let the autohinting algorithm do its magic by simply activating the *Autohint* option in the *File > Export* dialog. You can also enforce this setting with the *Autohint* custom parameter.

Test the hinting in an Adobe application (see 3.12.7, ‘Previewing in Adobe Applications’, p. 49). Write a test text and zoom out far enough so that the letters are displayed with a few pixels only. Then zoom in using the operating system’s *Zoom* function, which you can configure in the *Accessibility* settings of the System Preferences. If necessary, tweak your font settings or manually hint a problematic glyph and re-export. For details on manual hinting, see section 10.4, ‘Manual hinting’ (p. 126).

10.3.1 Flex Hints

If the font has cupped serifs or slightly tapered stems, the autohinter can automatically apply so-called flex hints. Flex hints suppress the display of such shallow curves at low resolutions. You cannot manually set flex hints, they are automatically applied when the font is exported. In order for flex hinting to kick in, a few conditions must be met.

First, the *blueShift* value must at least be set to the depth of your cups plus one. E.g., if your serifs are cupped 5 units deep, *blueShift* should be set to 6 or more. You can set *blueShift* as a custom parameter in *File > Font Info > Font* (Cmd-Shift-I).

Secondly, there are a few outline requirements. The cup or tapering must be built from exactly two consecutive outline segments between three nodes. The segments do not need to be symmetrical. The first and third node must share the same x coordinate (for tapered stems) or the same y coordinate (for cupped serifs). The four handles need not be completely horizontal (serifs) or vertical (stems), but the three nodes must

be placed on the extremes of the two segments. The overall depth must not exceed 20 units.

Thirdly, in the case of cupped serifs, it is recommended that the three points are completely submerged in the respective alignment zone. For best results, the second node (in the middle) should be exactly on the position (the 'flat edge') of the zone. And the other two nodes must reach into the zone. This means that cupped bottom serifs reach a little bit below the baseline, and into its bottom zone, which may seem counter-intuitive at first.

Flex hints: Nodes 1 and 3 are on the same level and inside the alignment zone, node 2 should be exactly on the flat edge of the zone. The handles must stay inside the space defined by nodes 1 through 3.



10.4 MANUAL HINTING

The implementation of PostScript hinting in Glyphs allows manual and automatic hints inside the same font. So, before resorting to manually inserting hints into your glyphs, we recommend to try to get as far as possible with autohinting. Only glyphs that do not display properly at low resolutions will need manual intervention.

Manual and automatic hinting cannot complement each other *inside the same glyph*. Any manually hinted glyph is excluded from the autohinting process. Thus, if you decide to add hints manually, you must *fully* hint the glyph.

There are two types of glyph-level hints you can add manually, *stem hints* and *ghost hints*. Stem hints describe a vertical or a horizontal stem or stem-like feature of a glyph, like a serif or a crossbar. Ghost hints mark top and bottom edges when a horizontal stem hint cannot be applied.

In combination with alignment zones, horizontal ghost and stem hints are important for the vertical alignment at the vertical font metrics, like the x-height or the ascender. At low resolutions, the rasterizer will try to vertically align the edges of all hinted horizontal stems that reach into an alignment zone. The horizontal hints must have their y coordinates in common with the nodes that are supposed to align. A single hint will do for all nodes it touches at its height.

Stem hints can overlap each other, e.g., the vertical stem hints in the figure eight. Technically, PostScript hinting does not allow overlapping of hints. So, in cases like this, Glyphs will automatically insert pieces of information called ‘hint replacement’, which turns hints on or off for different parts of the glyph outline. In practice, you do not need to worry about overlapping hints.

In a Multiple Master setup, only hints in the first master (the topmost master in the sidebar of *File > Font Info > Masters*) will be considered. In this case, make sure all your manually set hints are linked to nodes on the outline (see 10.4.1, ‘Stem Hints’, p. 127).

You can get a good start by right-clicking anywhere in the canvas in Edit mode, and choosing *Autohint* from the context menu. This way, glyph-level hints will be inserted similar to the way the autohinter would have done it when the font is exported. You can now edit them as described in the following sections. But keep in mind that, because of the presence of manually inserted hints, this glyph is now excluded from autohinting at export time.

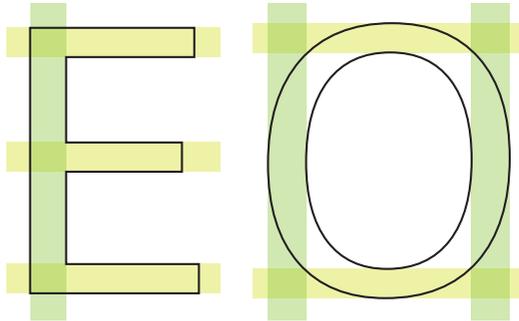
10.4.1 Stem Hints

To add a stem hint to a glyph, right-click and choose *Add Horizontal Hint* or *Add Vertical Hint* from the context menu. A gray bar with a number badge will appear. The numbers indicate the origin (first number) and width (second number) of the hint.

If you add a hint while two nodes are selected, the hint will be *linked* to these nodes right away. Adding linked hints this way even works on multiple node pairs at once, as long as each pair is on a separate outline. For best results, always

link your hints to extremum nodes (see 3.3.13, 'Extremes and Inflections', p. 26).

Positioning of vertical stem hints (green) and horizontal stem hints (yellow).



Select a hint by clicking on its gray number badge. Shift-click to select multiple hints. You can then edit the values numerically in the gray Info box (*View > Show Info*, *Cmd-Shift-I*). When exactly one hint is selected, press *Tab* to select the next hint, or *Shift-Tab* to go to the previous one.

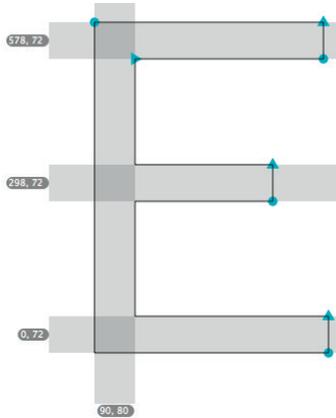
A horizontal stem hint, its number badge indicating a position of 140 and a width of 40.



To edit a hint graphically, drag the blue marks at the edges of the hint. The blue circle indicates the hint origin, while the triangle shows the size and orientation of the hint. If you drag one of the markers onto a node, Glyphs will link the hint to the position of the node. If you later move the node, the hint

will adapt accordingly. You can delete one or more hints by selecting them and pressing the Backspace or Delete key.

Hints linked to nodes with the blue triangle and circle. Stem hints must be positive, i.e., the triangle must be to the right or above of the circle.



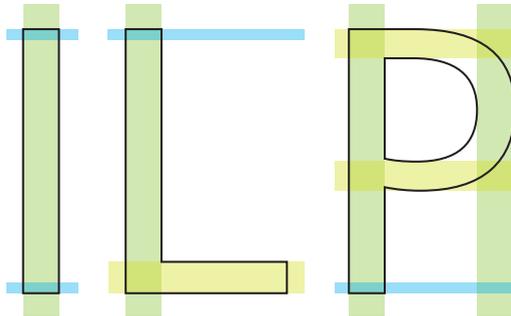
In the final OTF, all stem hints must be positive, i.e., have a width greater than zero. But even if you mistakenly insert a negative hint, Glyphs will correct its direction at export time, and all stem hints are turned positive.

10.4.2 Ghost Hints

You can use ghost hints when you need to vertically align the top or bottom of a glyph but cannot apply a horizontal stem hint. Take, for instance, a sans-serif uppercase I. The top needs to align with the cap height zone, the bottom with the baseline zone. In a serif I, you would apply horizontal hints to the serifs, but the sans-serif letter lacks the horizontal features necessary for a horizontal hint. In this case, you need to put a top ghost hint on the top of the I, and a bottom ghost hint at the bottom of the I. Similar situations occur on the top

of a sans-serif uppercase L, and at the bottom of a sans-serif uppercase P:

Positioning of ghost hints (blue) alongside regular stem hints (green).



You can create a ghost hint by right-clicking on a single node and choosing *Add Horizontal Hint* from the context menu. Turn any existing hint into a ghost hint by right-clicking the coordinate badge of a hint and choosing *Make Ghost Hint* from the context menu. The badge of a ghost hint only displays the position and its orientation. A downward arrow indicates a bottom ghost hint, an upward arrow a top ghost hint. Attach it to a point by dragging the blue circle onto a node. You can set its vertical orientation by selecting the hint and clicking on the upward or downward icon in the gray Info box (*View > Show Info*, Cmd-Shift-I).

A top ghost hint at position 100, and a bottom ghost hint at position 50.



10.4.3 Hinting Multiple Masters

In a compatible Multiple Master setup, you only need to insert hints in the *first* master. Provided the hints are linked to nodes on the outline, and the paths are compatible, they will be transferred to the corresponding nodes in compatible masters at interpolation time. The first master is whatever master is on top in the sidebar of *File > Font Info > Masters* (Cmd-I).

Manual hints in other masters will be ignored, unless there are no hints in the first master. Depending on the designs of your masters, it may make sense to hint a different master, or drag another master into the first position in the Font Info sidebar. For instance, if your first master is a very light weight,

it can become difficult to select the right nodes and apply manual hints on the outlines. In this case, it may be favorable to hint the Regular or Bold master.

And when you make use of Bracket and Reverse Bracket tricks (see 4.4.2, 'Special Layers', p. 53), you may also need to insert hints in the layer that replaces the master layer that carries manual hints.

11 TrueType Hinting

11.1 INSTRUCTIONING

Even though it is commonly referred to as ‘hinting’, including the title of this chapter, TrueType does actually not use ‘hints’. Rather, it applies something called instructions, which very precisely distort the outline for each pixel size. In other words, you do not give a ‘smart’ rasterizer hints about ‘dumb’ outlines, as in PostScript. On the contrary, in TrueType, you precisely instruct your outlines how to behave in different sizes, and the ‘dumb’ rasterizer just puts these ‘smartened’ outlines across the pixel grid. In practice, TrueType rendering results can still vary greatly between operating systems, versions of the same operating system, as well as between applications, and versions of the same application.

Instructioning improves the low-resolution rendering of TrueType-based fonts, such as OpenType/TT fonts (usually with a .ttf file name suffix), EOT webfonts (.eot), and TrueType-based WOFFs (.woff and .woff2). Like in PostScript hinting, the goal of the improved outline is not to preserve the glyph shapes, but rather to adapt them to the pixel grid. This can lead to immense outline distortions, especially in low resolutions.

TrueType fonts employ quadratic splines, i.e., outlines with different math for calculating curves, and a different path direction. But in the user interface of Glyphs, you always work in PostScript-style cubic splines. When you export to TTF, the paths are converted to TrueType on the fly, including all manually set instructions.

11.2 AUTOHINT

Glyphs employs the open-source TTFAutohint by Werner Lemberg for automatically adding TrueType instructions at export time. You can set autohinting options for each exported instance in *File > Font Info > Instances* (Cmd-I) by adding the custom parameter ‘TTFAutohint options’. For a detailed discussion of the available options, see its entry in the list of custom parameters in the Appendix (p. 197).

For more info on TTFAutohint, refer to the official website: freetype.org/ttfautohint

11.2.1 Either Manual or Automatic Instruction

Because of the far-reaching nature of the TrueType instruction code inserted into the font, the different approaches of

This is contrary to PostScript hints, in which some glyphs can have manual hints while other glyphs are autohinted.

TTF Autohint and the TrueType Instructor tool (shortcut I) are incompatible with each other. This means that you have to choose: You can have *either* manual instructions *or* automatic hints in an exported font, not both. Enabling TTF Autohint will override any manually set instructions at export time.

11.3 MANUAL INSTRUCTIONS

During rendering, pixels that fall inside the outline are on, pixels outside are off. What instructions can do, is distort the outline in a way that it produces a better low-resolution picture when laid across the pixel grid. Thus, all instructions, in one way or another, move outline nodes relative to the underlying pixels.

Glyphs supports four kinds of TrueType instructions: Align, Stem, Interpolation, and Diagonal. In order to insert these instructions, you need to switch to the TrueType Instructor tool (shortcut I), select a certain number of nodes, and choose an instruction from the context menu. You can access the context menu by either right-clicking or Ctrl-clicking anywhere in the canvas.

No matter what the selection is, the context menu of the Instructor tool always provides the two options *Autohint* and *Remove All Hints*. *Autohint* attempts to guess which instructions are appropriate for the current glyph and inserts them. You achieve better results if stems and zones are set properly. *Remove All Hints* deletes all instructions from the current glyph. You can achieve the same effect by selecting all instructions (Cmd-A) and deleting them by pressing the Backspace or Delete key. A node that changes its position because of an instruction is considered 'touched', while a yet unmoved node is referred to as 'untouched'.

In Multiple Master setups, only instructions in the first master are considered, and, assuming the masters are compatible, automatically applied to all interpolated instances.

You can keep your overlaps, and even hang instructions to nodes inside an overlap, wherever necessary. When overlaps are removed at export, the TrueType instruction in the overlap is adapted as well, and moves to the resulting intersection node, provided that the node is close to the intersection. This means that, in order to make a hint work as expected in an overlap, you may need to either move the touched nodes closer to the intersection, or remove the overlap altogether.

When manually instructing glyphs, first focus on the vertical alignments of the whole glyph, i.e., its baseline alignment, the glyph height, etc. Then, take care of all vertically measured features of your glyphs, like the thicknesses of serifs and crossbars, and counter heights.

Ignore horizontally measured attributes, like vertical stems, because these are already taken care of by modern subpixel rendering methods like Microsoft's ClearType. The TrueType Instructor tool only applies horizontal hints, i.e., hints for vertically measured attributes described in the previous paragraph.

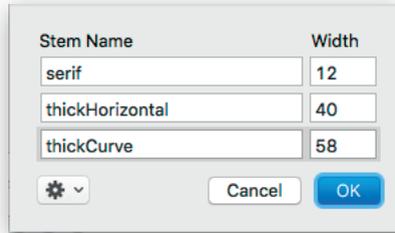
11.3.1 Horizontal Stems and Zones

The TrueType instructions reuse the alignment zones set in *File > Font Info > Masters > Alignment Zones* and the standard stems in *File > Font Info > Masters > Horizontal Stems*. See section 10.2, 'Font-Wide Hints' (p. 122) for details.

However, TrueType easily supports more stems than PostScript. You can set separate horizontal TrueType stems with the *TTF Stems* custom parameter in *File > Font Info > Masters > Custom Parameters*. After adding the custom parameter, access its options by clicking in its *Value* field. Via the gear menu of the following dialog sheet, you can import existing horizontal PostScript standard stems, define additional stems, and give the stems descriptive names. In Multiple Master setups, make sure that all masters contain compatible *TTF Stems* parameters. That means that all parameters need to have the exact same number, order and naming of stems. You can copy and paste parameters between the *Custom Parameters* fields of the individual masters in *File > Font Info > Masters*.

Finding good standard stem values works very much like finding good average values in PostScript hinting, except that it is safe to add more stem values. You can even add stem entries for horizontal stems that appear only rarely in the font. That is because you can specifically apply a named stem to a stem hint later. It is advisable to have well-differentiated standard stems. I.e., only add an additional horizontal stem if it is different enough from existing stems, because different standard stems will cause diverging pixel thicknesses in some

font sizes. Only of this is acceptable for your design, you should make the differentiation between two stems.



11.3.2 Rasterizer Preview

While you edit a glyph with the TrueType Instruction tool, you will see a red contour behind the glyph outline. This is a precise preview of the instructed outline, which depends on (a) the instructions you add to the glyph, (b) the font size, and (c) which rendering intent is set. Three different rendering intents are available for preview: Grayscale, GDI ClearType, and DirectWrite. Additionally, Glyphs draws a pixel preview of the currently selected instance in the background. However, the pixel preview is only a vague reference. Actual display renderings will vary in different applications and environments.

Tip: To preview the instructed outline and the pixel rendering of other instances than the current one, select a different instance in the Preview area at the bottom.



From left to right:
4×4 oversampling in Grayscale,
8×1 oversampling in ClearType,
5×5 oversampling in DirectWrite.

Grayscale rendering applies four times horizontal and four times vertical oversampling for each pixel. This means that every pixel is subdivided into (4×4=) 16 pixel parts. Depending on how many of the centers of these subdivisions are located within the instructed outline, one out of 16 grayscale values will be used for drawing this pixel on the screen. GDI ClearType, mainly used in Windows XP, applies 8×1 oversampling, i.e., 8 times horizontally, but no oversampling vertically. DirectWrite, in use since Windows 7, makes use of 5×5 oversampling.

You will see a difference in outline snapping when you switch between these modes. While the instructed outline can snap to pixel fourths in Grayscale mode, and pixel fifths in DirectWrite, it will only snap to full pixel heights in ClearType.

In the gray Info box (Cmd-Shift-I), you can use the drop-down list to switch between the three rendering modes Grayscale, ClearType and DirectWrite. In the size chooser next to it, you can switch between point sizes in PPM.

Windows assumes a screen resolution of 96 ppi. To calculate the actual PPM sizes, divide the Windows point size by 3 and multiply it by 4, e.g., a Windows font size of 12 points corresponds to $12 \div 3 \times 4 = 16$ pixels on the screen. Likewise, to calculate the Windows point size, multiply the PPM size by 3 and divide by 4, e.g., a PPM size of 20 pixels corresponds to a Windows point size of $20 \times 3 \div 4 = 15$ points.

11.3.3 Anchor

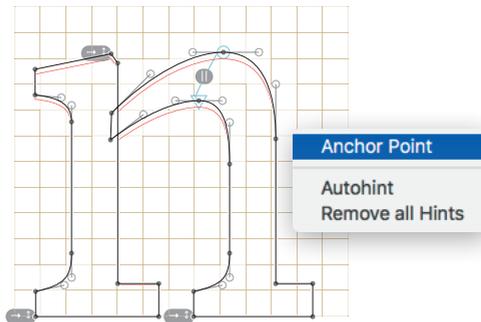
To insert an Anchor instruction, select one node, preferably in an alignment zone, right- or Ctrl-click to bring up the context menu and choose *Anchor*, or press the shortcut A. The point must not already be touched by another instruction, such as a Stem or an Interpolation.

The Anchor instruction moves a node up or down to the nearest pixel edge. Its direction is indicated by a small blue triangle pointing up or down. To toggle the direction of the Anchor, select it by clicking on its badge, and click on one of the indicators which appear in the gray Info Box:  round to nearest edge (default option),  round up,  round down, or  do not round.

Tip: The TT instructing shortcuts are the letter keys A through G in the second row of a U.S. keyboard. On a French keyboard, for instance, you would have to press Q instead of A.

The shortcuts override the tool shortcuts. This means that you cannot access the Annotations tool (default shortcut A) while the TT Hinting Tool is active.

On the top serif and on both bottom serifs, three Anchor instructions are used to snap the ends of all stems of this lowercase n onto the pixel grid.



You do not need to anchor a node if it is going to be the origin of a Stem or Align hint, because Glyphs will automatically (and invisibly) insert an Anchor instruction on that node.

11.3.4 Align

Select two nodes, then choose *Align* from the context menu to insert an Align instruction, or press the keyboard shortcut F. The Align instruction binds an (untouched) target node to a

(touched) origin node, so that the target node reduplicates the shift of the origin node.

If you apply an Align instruction on two untouched nodes, the origin node will automatically (and invisibly) receive an Anchor instruction. Typically, the node inside a zone will become the origin. You can change the direction of the Align instruction by selecting it and choosing *Reverse* from its context menu.

Align instructions can be useful if you want to replicate the shift caused by a stem hint in another node. E.g., you apply a stem hint to a left serif, but want the node on the opposing right serif to travel just as far. In that case, you select the moved node of the left serif and the yet unmoved node of the right serif, right-click and choose *Align* from the context menu.

11.3.5 Stem

Select two, four or six nodes, and choose *Stem* from the context menu to insert one, two or three stem hints. Alternatively, you can press the keyboard shortcut S. The blue circle represents the origin of the instruction, and the triangle stands for the target. Glyphs will automatically insert an invisible Anchor instruction on the origin node, unless it is already targeted by another TrueType hint, such as an Interpolate instruction.

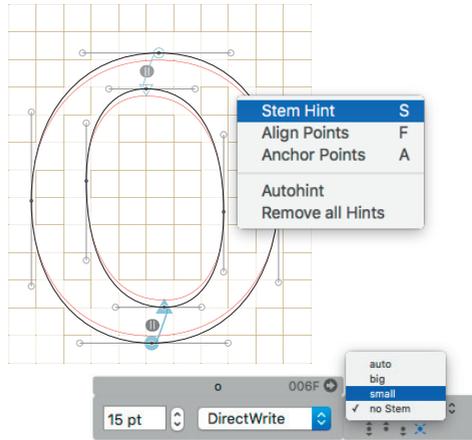
Stems (a.k.a. 'links') control the vertical distance between two nodes. The origin point of the stem will be snapped to the nearest oversampling edge, or to the nearest pixel edge if the point lies within a zone or has been anchored. You can round the stem onto the pixel edges. To do that, select it by clicking on the badge of the Stem, and click on one of the indicators which appear in the gray Info box:  round to nearest edge (default option),  round up,  round down, or  do not round. The target point will be positioned in relation to the origin point. The distance between the origin and target points is linked to a horizontal stem value defined in the *TTF Stems* parameter in *File > Font Info > Masters > Custom Parameters*. For details, see its entry in the list of custom parameters in the Appendix (p. 198). To change the stem the instruction is linked to, select one or more Stem hints by (Shift) clicking on their badges and choose a stem from the drop-down list that appears in the gray Info Box. You can set it to *No Stem* to reduplicate the shift of the source point in the target point.

Tip: Avoid rounding if you want to profit from oversampling. Since all rounding rounds to full pixel edges, this mainly makes sense for rendering intents lacking vertical oversampling, i.e., for GDI ClearType on Windows XP.

Thus, a Stem hint with the *No Stem* setting is equivalent to an Align instruction.

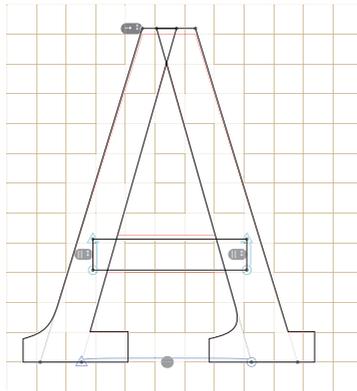
Two Stem hints have been placed on the top and bottom of the lowercase o. The lower Stem hint is selected, and a TTF stem is applied in the gray Info box.

That is why the lower stem snaps to an oversampling edge for the selected rendering intent.



Right-click or Ctrl-click the Stem hint, and choose *Reverse* from the context menu in order to change the direction of the instruction. We recommend to put Stem instructions from points in the alignment zones towards the center of the glyph, as well as on crossbars as in A, H, or f. Serifs can also be homogenised with Stem instructions.

In this uppercase A, two Stem hints on both sides of the crossbar preserve the diagonal angles of the vertical stems.



If you place equal Stem instructions on both ends of a crossbar, Glyphs will try to preserve angles of intersected diagonals. In this case, the points will not just be moved up or down, but in the direction of the surrounding diagonal

segments. This can be useful, for instance, in an uppercase A or an italic uppercase H.

11.3.6 Triple Hint

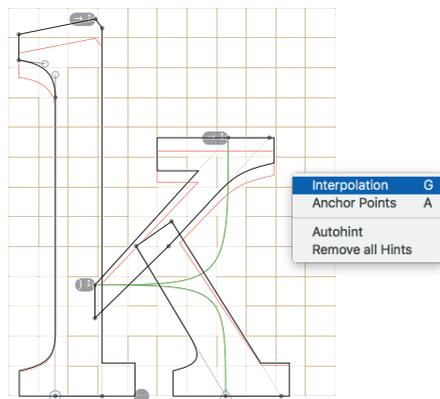
Select three stem hints by subsequently Shift-clicking on their badges, then right-click and choose *Make Triple Hint* from the context menu. Triple hints will link the size of the stems to each other, and reduce them if necessary, i.e., if there is not enough vertical space to accommodate all three stems. Thus the stems will be displayed with fewer pixels than the *TTF Stems* parameter would indicate, but the counters are preserved. This is useful in dense letters such as the lowercase a and e, or the small cap B and E.

11.3.7 Interpolate

Select three nodes, and choose *Interpolate* from the context menu, or press the keyboard shortcut G, to insert an interpolation instruction. Two of the nodes must be touched by other instructions.

An Interpolate instruction will move a point relative to the movement of two other, touched points. This way, you can make sure that the relative positioning of the third node is preserved when instructions are applied to the outline. This is especially helpful for balancing counters and middle strokes, as found in letters such as B, E, K, and R, or the crossbar of an uppercase G:

In this lowercase k, an Interpolation hint determines the node on the vertical stem by interpolating the vertical position from two touched points: an aligned node on the leg and an anchored node at the end of the arm.

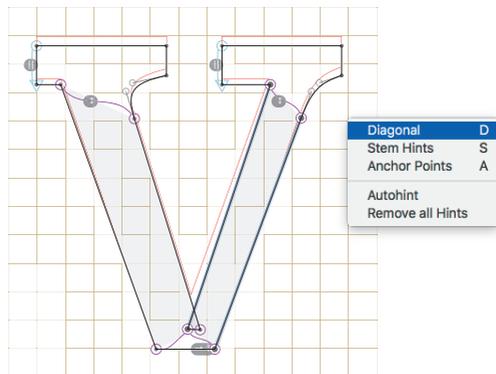


11.3.8 Diagonal

Subpixel rendering in GDI ClearType allows verticals and diagonals to grow by partial pixels, while horizontal widths grow by full pixels. So, when scaling through a range of PPM sizes, diagonals will appear too thick right before the full-pixel increase of horizontals, and too thin right after a full-pixel increase.

A Diagonal instruction tries to replicate the jumps in thicknesses of a specified horizontal stem. Thus, the weight relation between diagonals and horizontals can be preserved. To apply a Diagonal instruction, select four nodes, and choose *Diagonal* from the context menu, or press the keyboard shortcut D. Link it to a horizontal stem width in the glyph's Info Box.

Two Diagonal hints are used to normalize the two stems of this lowercase v.



This is useful in all glyphs with prominent diagonals, such as k, v, w, x, y, and z.

11.3.9 Automatic Interpolation of Untouched Points

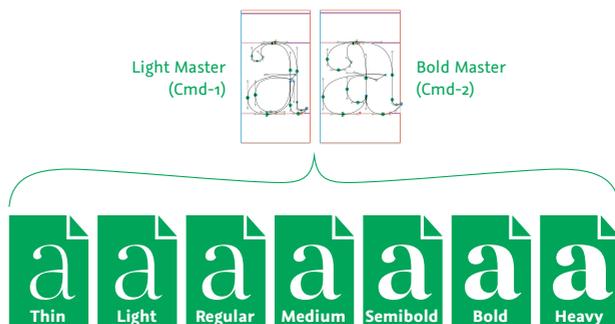
The position of remaining untouched nodes will be interpolated between the positions of the touched nodes. This automatic interpolation is applied after all instructions have been executed, and the positions of all touched nodes have been determined. This means that you can usually achieve the desired outline distortion with placing a minimum of instructions, first on vertical extremes, and then, if necessary, working you way inward.

12 Multiple Masters

12.1 OVERVIEW

You can draw two or more extreme styles of a font family, so-called ‘masters’, e.g., the Light and the Bold. If you keep them in one single file, this is referred to as a Multiple Master setup. Such a configuration allows you then to define ‘instances’ in between, i.e., fonts calculated by the software. In other words, you draw the masters, the input. And Glyphs outputs the instances for you.

A small number of masters can be used to interpolate a large number of instances. In this example, two masters on a Weight axis (Light and Bold Master) are interpolated to produce seven font instances, from Thin to Heavy.



You can put two or more masters on one ‘interpolation axis’, i.e., an imaginary line between the masters, along which a style feature can be interpolated. The two most common axes are Weight (i.e., from Light to Bold) and Width (i.e., from Condensed to Extended), but you can interpolate any other design feature as well, be it contrast, serif thickness, ascenders and descenders, or whatever makes sense for the design in question. Glyphs supports up to three axes: Weight and Width are predefined, and a third custom axis, which you can name yourself.

The relation between the masters is defined by their Width, Weight, and custom values. Utilizing these values, the masters span a design space inside of which the instances can be placed. It is also possible to extrapolate, i.e., place instances outside the design space. But in practice, extrapolation proves to be very error-prone and thus not recommended, unless you prepare your masters in a way fit for extrapolation.

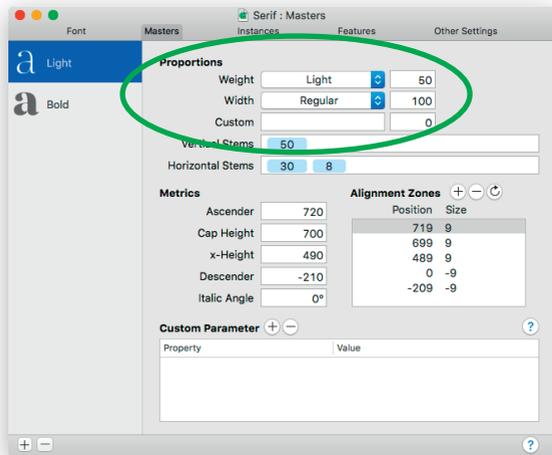
Layers are independent. That means that the nodes, handles, and anchors are not automatically in sync between the masters. So you can insert nodes in one master while all

Tip: It is advisable to keep uprights and italics in different files since their basic shapes are too different to linearly interpolate between them. See section 12.6, ‘Ensuring Family Consistency across Files’ (p. 148).

other masters stay untouched. Note that for the interpolation to work, the masters need to be compatible. That means that every node, handle, anchor, and component in one master must correspond to the same structure of nodes, handles, anchors and components in all other masters involved in the interpolation. Incompatible masters will cause the interpolation to fail, and will yield an empty glyph in the exported font.

12.2 SETTING UP MASTERS

Add masters in *File > Font Info > Masters* (Cmd-I), by clicking on the plus button in the bottom left corner of the window. For a detailed description of the options in this window, see section 7.2, ‘Masters’ (p. 86). Defining the coordinates of the masters sets up the design space. For the *Weight* coordinate, we recommend using the stem width of a representative letter such as the lowercase n. This makes it easier to define the stem widths of the instances. The numeric *Weight* coordinate is entered in the text entry field next to the pop-up, all the way to the right:



Use the *Weight* pop-up to choose a master name. The names are presets, and only influence the naming of the master layers in the Palette, and the icon representing the master in the toolbar of the main window. In other words, this setting is not used for the style of the exported font (i.e., the instance).

The style names of the final fonts are defined in *File > Font Info > Instances*.

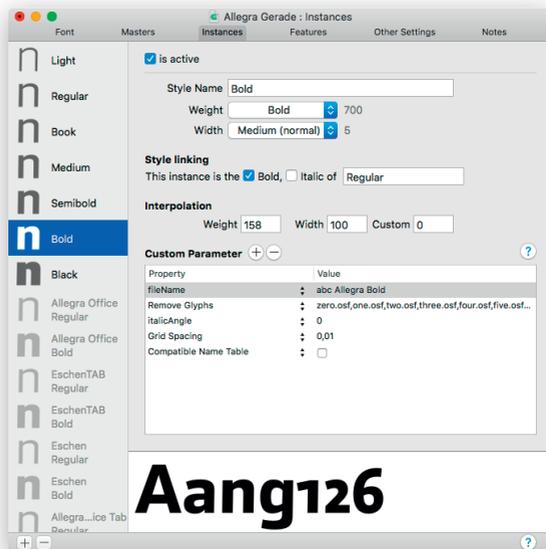
If you need a second or third dimension for your interpolation, you can add a *Width* setting and / or a *Custom* third axis, where you can type your own name in the Custom text field. The name of the master will be composed of the names chosen for the *Weight*, the *Width*, and the *Custom* axis.

Keep an eye on the interpolation values in the number fields on the right. Every master must have distinct values there, otherwise interpolation of instances is not possible.

You can re-order masters by dragging their entries in the sidebar into a new position. The order of the masters does not affect interpolation, though the first master does have significance for a number of other functions, especially hinting, but also color and layered fonts.

12.3 SETTING UP INSTANCES

Once you have defined the design space with the values of the masters, you can pick instance positions relative to the masters. If you keep instances between masters, you are setting up an interpolation. Instances placed outside the masters are extrapolated. You manage instances in *File > Font Info > Instances*. For a detailed description of all options in the window, refer to section 7.3, 'Instances' (p.90).



Set a *Style Name*, and make sure it is unique for the font family, otherwise fonts will be overwriting each other when exported. Then, pick appropriate *Weight*, *Width*, and *Custom* settings. The values correlate to the *Weight*, *Width*, and *Custom* values of the masters you have set up. If you keep instances between masters, you are setting up an interpolation. Instances placed outside the masters are extrapolated.

12.4 FIX OUTLINE INCOMPATIBILITY

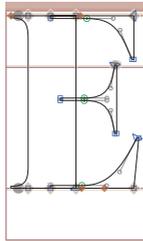
When merging masters, Glyphs does not automatically fix incompatible outlines. This has to be done manually. Letters with incompatible outlines are marked with a pale red triangle in the Font view and a red top stripe in Edit view.

In this example, the uppercase E is marked as incompatible in Font view. Note the pale red triangle in the upper left corner of its cell.



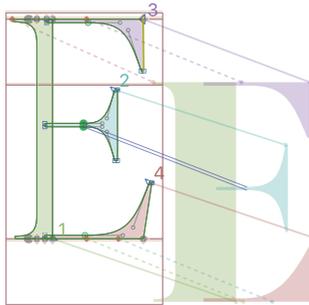
During editing, it is possible to work with incompatible paths, but for interpolation, they need to be compatible. Specifically, this means that the order of the paths as well as number and structure of nodes must correspond with each other.

When active in Edit view, an incompatible glyph is marked with a pale red stripe on top of the Ascender, and across its width.

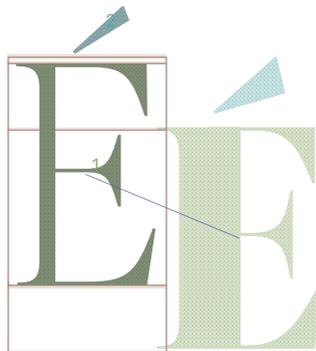


View > Show Master Compatibility (Ctrl-Opt-Cmd-N) activates a visualization of the masters' congruency in Edit view. All masters are displayed in an exploded view. Paths are colored and numbered according to their order in the glyph layer. Starting points of corresponding paths are connected with solid lines in the color of their path. Corresponding anchors are connected with colored dashed lines. If a path or an

anchor is missing in a master, the solid or dashed line will point to the origin point of that master. Selected nodes will be connected to their corresponding nodes in other masters with solid blue lines.



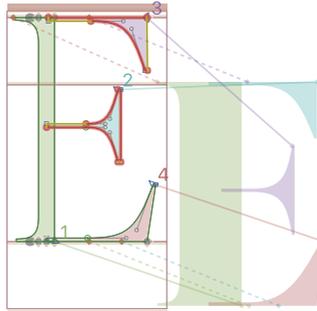
Components are displayed in a checkered pattern, colored and numbered according to their order. And, if selected, they are connected with a solid blue line from center to center. If a component is missing in one master, the indicator line will connect to the origin point.



The various outline segments are matched up against each other and receive a subtle color overlay. Segments marked red are technically incompatible, i.e., impossible to interpolate. This usually means that a line segment is matched up against a curve segment, which is not possible to interpolate. Segments marked in green are technically compatible without problems. Segments marked in yellow are technically compatible, but their orientation changes between masters,

e.g. from straight to slanted, or from slanted to the left to slanted to the right. Yellow markings can also be a false alarm, especially if the rest of the outline is marked green. Sometimes, however, this points to a badly placed start point, resulting in a technically feasible, but unwanted interpolation.

While the vertical stem on the left interpolates just fine and is therefore marked green, the upper and middle serifs are not in the same order in both masters, resulting in some segments marked red and some yellow.



All masters of your glyph that interpolate need to be compatible in terms of paths, components, and anchors.

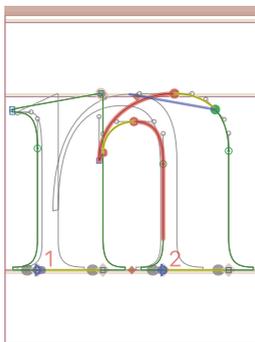
If your paths are not compatible, it is usually a good idea to first run *Paths > Correct Path Direction* (Cmd-Shift-R) or, while holding down the Option key, *Paths > Correct Path Direction for All Masters* (Cmd-Opt-Shift-R). Glyphs will then try to reorder the paths, reset the starting point, and fix the path direction. If the node counts are not the same across master layers, you can add nodes to the outline with the Draw tool (shortcut P), or delete nodes by selecting them with the Select tool (shortcut V), and subsequently pressing the Delete key.

You can rebuild compounds to their default recipe with *Glyph > Make Component Glyph* (Cmd-Opt-Shift-C). If you are employing a non-default component structure, you can simply copy and paste components between layers. For more about components, see section 8.1, ‘Components’ (p. 98).

To change the order of paths and components, use *Filter > Fix Compatibility*, and rearrange the objects in the subsequent dialog. For more details, see section 5.2.1, ‘Fix Compatibility’ (p. 57). To set start points, Ctrl- or right-click a node and choose *Make Node First* from the context menu.

Glyph > Set Anchors (Cmd-U) will add all missing default anchors on the current layer. Holding down the Option key, you can delete all existing anchors and set only the defaults with *Glyph > Reset Anchors* (Cmd-Opt-U).

If you prefer a stacked view to the exploded view for *View > Master Compatibility* (Ctrl-Opt-Cmd-N), try deactivating *Master Compatibility with Offset* in *Glyphs > Preferences > User Settings*. In that case, paths and components will not be colored according to their order, but be numbered with red figures next to their start points. This view makes it easier to compare heights and overshoots.



12.5 COMPARING MASTER LAYERS

The quickest way to look at different master layers is to switch between them by either using the tool bar buttons or the corresponding shortcuts, Cmd-1, Cmd-2, etc., i.e., the Command key and the number of the respective master.

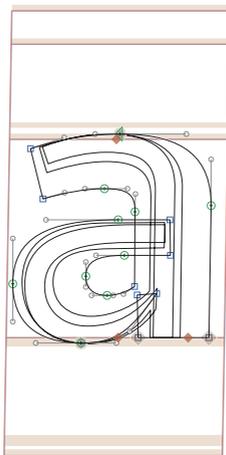


The *Layers* palette (*Window > Palette*, Cmd-Opt-P) shows all layers of the selected glyph. The master layers are shown in bold. When clicking an entry in the list of layers, the selected layer is activated in the current glyph only. If multiple glyphs are selected, you can still pick any of the (bold) master layers. This is useful for comparing different masters for one or more glyphs side by side.



If you want to see all master layers of the current glyph side by side in Edit view, choose *Edit > Show All Masters*. This also inserts Brace or Bracket layers, if present.

Using the visibility icon (the eye symbols) in the *Layers* palette, you can have Glyphs display the paths and components of the activated layers as gray outlines on top of each other:



12.6 ENSURING FAMILY CONSISTENCY ACROSS FILES

If you extend your font family beyond the scope of a single Glyphs file, e.g., with italic styles, then make sure both files carry precisely the same *Family Name* in *File > Font Info > Font* (Cmd-I). Keep in mind that Font family names can be altered for an instance with a custom parameter. Except for a Bold Italic, italic styles always need to be style-linked with their non-italic counterpart in the Instances tab. The Bold Italic needs to be style-linked to the Regular. See section 7.3.4, ‘Style Linking’ (p.92), for more details.

To compare the glyph scope and glyph metrics of two or more open Glyphs files, choose *Edit > Compare Fonts*. Glyphs will then display a spreadsheet containing glyph counts, an indication which glyphs are missing in which file, and a per-glyph comparison of LSB, RSB, and width values for each Master layer. If you want to add the missing glyphs to one of the fonts, you can copy the names of the missing glyphs from the spreadsheet, and paste them in the dialog of *Glyph > Add Glyphs* (Cmd-Shift-G).

12.7 BRACE LAYERS

If a layer name consists of, or ends with curly braces {...}, and the braces contain comma-separated number values, representing coordinates in the Multiple Master design space, it is treated as a so-called 'Brace layer'. Paths, components, and anchors on a Brace layer will act as an intermediate master for the respective glyph only. The layer name can either contain one value for a single-axis weight interpolation, two values, representing weight and width interpolation values, or three values for a three-axis setup. Brace layers can be useful for fine-tuning the interpolation of horizontals in dense glyphs such as e or s.

Top row: without Brace layer.
Bottom row: with Brace layer
in the middle. Note the growth
of the crossbar: linear in
the first half, proportionate
in the bolder weights.



To quickly set up a Brace layer in a glyph, duplicate a master layer by clicking on the *Copy* button in the *Layers* palette, then rename it to, e.g., {100} for a single-axis setup, or {100, 90} or {100, 90, 80} for two- or three-axis setups, the numbers representing the design space coordinates on the Weight, Width and Custom axes. Make sure the layer is selected, then choose *Re-Interpolate* from the gear menu in the *Layers* palette. Now, adjust the outlines on the Brace layer. You can see the effect of your changes immediately if the Preview area at the bottom of the Edit view is set to *Show All Instances*.

12.8 BRACKET LAYERS

If a layer name starts with the name of a Master, followed by a space and brackets [...] containing exactly one number value, the layer is treated as a Bracket layer. Bracket layers exchange the Master indicated by the name at or beyond the interpolation value indicated between the brackets. E.g., 'Bold [130]' will replace the Bold master with the contents of the Bracket layer beyond a weight of 130. You can have any number of Bracket layers, i.e., exchange any master as often as you like. You can exchange both masters at the same time,

e.g., with 'Light [100]' and 'Bold [100]'. In a two-axis setup, you can add an additional threshold value for the Width axis with a comma, e.g., 'Bold [100,300]' will replace the Bold master if both the weight value is beyond 100 and the width value exceeds 300.

Bracket layers are an easy means of switching shapes for glyphs that look substantially different between thin and bold weights. A common usage example is the vertical bar that crosses down through the dollar sign. In bolder variations of the glyph, most designers choose to split it in two and reduce it to whatever extends below and above the s-shaped part of the sign. Keep the light variations as masters and put the bold variants into the bracketed layers:



Alternatively, you can keep a separate, non-exporting dot variant of the glyph, e.g., dollar and dollar.bold, and employ the *Rename Glyphs* parameter in the bolder instances for replacing the regular shape with the dot-suffixed glyph. Some people prefer this method over Bracket layers because this way, alternate masters are always visible in Font view.

12.9 OPEN BRACKET LAYERS

An Open Bracket layer is a layer with a name of the structure *Mastername [weightvalue]*, i.e., the name of the master, followed by a space, a right bracket, a weight interpolation value, and another right bracket, e.g. 'Bold]100]'. It acts very much like a normal Bracket layer, except that the alternate master that is being swapped in at the specified weight value, lies already in the Master Layer, while the Open Bracket layer contains the master that is active until that point. The advantage of an Open Bracket layer is that you always see the correct master in Font View.

13 Color Fonts

Glyphs supports a streamlined workflow, preview and support of four types of color fonts: classic layer fonts, Microsoft-style fonts with CPAL and COLR OpenType tables, Apple-style fonts with an sbix OpenType table, and Mozilla/Adobe-style fonts with SVG tables. All of them have in common that you place different content on multiple layers.

13.1 WORKING ON MULTIPLE LAYERS

13.1.1 Select All Layers Tool

For editing vectors on multiple layers at the same time, you can pick the Select All Layers tool. It works much like the Select tool, except that it is effective on all visible layers at the same time. You can switch between the Select and Select All Layers tools by clicking and holding the icon, then picking the tool you want from the menu that pops up. Alternatively, press Shift-V to toggle between the two selection tools.

13.1.2 Keeping the Metrics in Sync

Since the shapes on the various layers need to keep their positioning with respect to each other, both glyph widths and kerning must be shared across all involved layers. To facilitate this, you can add a custom parameter called *Link Metrics With First Master* to *File > Font Info > Masters*. Subsequently, all master and color layers will follow all metrics changes in the first master.

13.1.3 Exporting Color Fonts

All color fonts are regular OpenType fonts. They work in any format Glyphs can export, be it OTF/CFE, TTF, WOFF, WOFF2, or EOT. However, they will only work in environments that allow their display. For instance, layered fonts will only work in a software that can put pieces of text exactly on top of each other, like most DTP software. This is very hard or impossible to achieve in word processing apps such as TextEdit or Microsoft Word.

13.2 LAYERED COLOR FONTS

Layered color fonts are simply separate fonts that can be stacked on top of each other. So, the software environment in which the font is supposed to be used, must support the stacking of text on top of each other.

13.2.1 Named Masters with Master Colors

In order to draw on different levels, you need to add a master for each level in *File > Font Info > Masters*. You can give them custom names in the *Custom* field. Each custom-named master must have a unique value in the number field next to the name field. E.g., one master called 'Front' with value 1, and one master called 'Side' with value 2.

To assign a preview color to a master, add the custom parameter *Master Color* to the respective master. Set the color with the color picker that appears after you click on the color field in the parameter value. Keep in mind that the color is only intended as preview color within Glyphs.

13.2.2 Instances for All Masters

In order to set up individual font files for the respective color layers, go to *File > Font Info > Instances*, then click on the plus menu in the lower left of the menu, and pick *Add Instance for each Master* from the menu that pops up. Glyphs will then add instances that exactly match the masters you have set up in *File > Font Info > Masters*.

13.2.3 Previewing and Using Layered Color Fonts

You can switch between masters just as you would in any other multiple master setup. To preview various masters at the same time in Edit view, switch to the Text tool (T) or hold down the space bar. All master layers that are set to show in the *Layers* palette (Cmd-Opt-P) will be displayed in their respective master colors as set in the custom parameters in *File > Font Info > Masters*. For more info on toggling the display state of a layer, see section 4.4, 'Layers' (p. 52).

Be prepared that the users of your layered font will also want to use it in Adobe Illustrator. In this application, the default baseline offset for the first line in a text box is calculated as the height of the bounding box in the lowercase d. In other words, the highest nodes in your lowercase d will be aligned to the top edge of the text box.

Since it is unlikely that the color layers happen to have the exact same height *d*, layered color fonts will appear misaligned in Adobe Illustrator. In order to alleviate the problem, you can either instruct your users to change the default setting in *Type > Area Type Options > Offset > First Baseline* from *Ascent* to *Em Box Height* or *Fixed*. Or, you can add a tiny closed path (e.g. a one-unit square) to the top of the lowercase *d* in all color masters.

Rendering of a layered color font in Adobe Illustrator with the default setting for baseline offset (above) and set to Em Box Height (below).



13.3 MICROSOFT COLOR FONTS

Microsoft-style color fonts are fonts that contain two additional tables: CPAL (color palette) and COLR (color). The CPAL table indexes an arbitrary set of colors, and the COLR table describes which color index is applied to which outline.

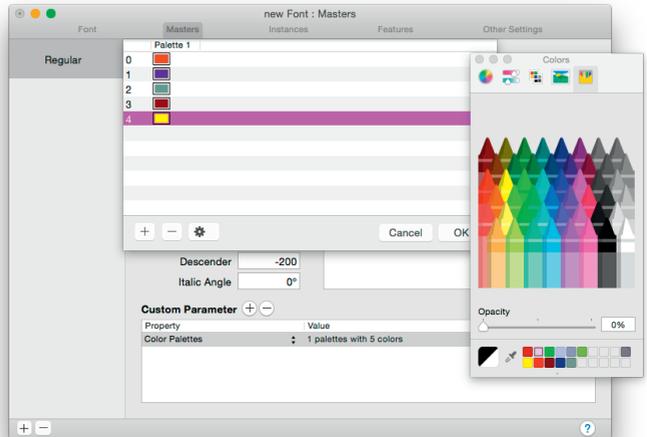
13.3.1 CPAL Table

In order to set up a color palette, go to *File > Font Info > Masters* and add a custom parameter called *Color Palette*. Click in its Value field to bring up a dialog sheet for setting a color palette. The dialog displays one palette per column, and a color index per row. The leftmost column shows the index number, starting at zero.

In the dialog sheet, you can add new colors with the plus button, and remove existing ones with the minus button. To edit a color, click to select a color, and then click again on the color field. Or, simply double click any displayed color field. The system color picker will appear, allowing you to choose a new color value for the color index in question.

The gear menu allows you to add or remove additional palettes. Additional palettes may make sense for color variations, e.g., for a light and a dark background. However, Glyphs will only preview the first palette for you, and at the time of this writing, no software was known that was able to access other palettes than the first one.

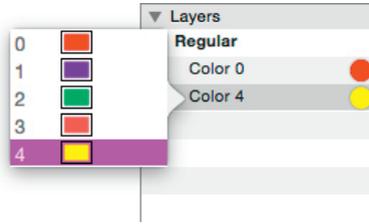
Once you are done, click OK to confirm the color choices, and a palette is inserted into the font.



13.3.2 Creating Color Layers

In order to apply the colors from the color palette, you can set up any number of glyph layers that follow the naming scheme 'Color<space><color index>', e.g., 'Color 2' for the third color. Once such a color layer is set up properly, a color marker will appear next to the name, indicating the color. You can have several layers sharing the same color. You can change a layer to a different color either by renaming the layer accordingly, or by clicking on the color marker, and choosing a different color from the menu that pops up either by clicking on it with the

mouse pointer or by selecting it with the up and down arrow keys and confirming your selection with the Return key.



13.4 APPLE COLOR FONTS

Apple's color fonts carry a table called `sbix`, which contains bitmap data for various resolutions. The `sbix` table is currently only supported in OS X versions 10.7 and later and iOS. Some functionality requires at least OS X 10.9 or iOS 7. Fonts containing an `sbix` table currently cannot be displayed in Adobe applications or on Windows. More technical information the `sbix` table can be found on:

- developer.apple.com/fonts/TrueType-Reference-Manual/RM06/Chap6sbix.html

13.4.1 Bitmap Image Files

Since Glyphs is not a binary image editor, you have to prepare the pixel images with another application such as Preview, Pixelmator or Photoshop. Also keep in mind that in the Glyphs file, only the relative paths are stored. We therefore recommend to keep all images in a folder next to the Glyphs file. And when you move the file, always move the folder along with it.

The following file formats are supported: PNG, JPEG, and TIFF. PNG transparency is supported as well. The spec also allows PDFs, but at the time of this writing, embedded PDFs are not displayed in any software known to us.

You can prepare different images for different pixel sizes, e.g., one for 24 pixels, one for 48 pixels, and one for 512 pixels. The application displaying your font will pick the right size variant for the font size in question. If you choose to have different sizes, make sure that you prepare each of your images for all intended sizes.

13.4.2 Resolution Layers

In order to insert color bitmaps into your font, create additional layers in your glyphs and name them according to this naming scheme: ‘iColor<space><image size>’, e.g., ‘iColor 48’ for the 48-pixel tall image, or ‘iColor 512’ for the image with a height of 512 pixels. Once you have set up one or more iColor layers, simply drag the images from the Finder into the respective layers. Alternatively, you can select the layer, choose *Glyph > Add Image*, and pick the respective image file in the dialog sheet that appears.

Once the image is placed on the layer, the iColor layer is finished. Neither position nor scale of the placed image matter. So you can leave it at its placement next to the origin point of the layer. When displayed, the image will be scaled to the pixel size on screen, and moved vertically so that approximately 15% of its height is below the baseline. The glyph will inherit its width from the master layer. You can use the master layer for a normal outlined drawing. It will usually be displayed on top of the scaled bitmap image.

Make sure to keep the resolutions in sync across all glyphs that employ sbix images. I.e., if you set up one glyph with iColor 64, iColor 256, and iColor 512, then all other color glyphs must have the same set of layers (and their respective placed images).

13.4.3 Using Apple Color Fonts

While an sbix table can be placed in any OpenType-based font, its use on the web is currently limited to Safari on Mac OS X 10.10 and later and font formats supported by it, i.e., OTF, TTF, and WOFF. Previous versions of Safari will only display the sbix color glyphs if they come from an OTF or TTF that is installed on the user’s machine. Installed locally, it also works in all apps using the Cocoa text engine, e.g., in TextEdit.

13.5 SVG COLOR FONTS

SVG Color Fonts must not be confused with the SVG webfont format (i.e., fonts with a ‘.svg’ or ‘.svgz’ file name suffix), which is not supported by Glyphs. Rather, SVG Color Fonts are regular OpenType fonts with embedded SVG graphics. This is done through the means of an SVG table in the OpenType font file. SVG Color Fonts were suggested by Mozilla as a standard for color fonts and subsequently backed by Adobe. Therefore, they

You can read the spec at w3.org/community/svgopentype/

are also referred to as ‘SVG-in-OpenType’ or ‘Mozilla/Adobe-style colorfonts’.

The feature that sets SVG Color Fonts apart from other color font solutions is that it also allows the embedding of animations. Currently, only Firefox versions 26 and later support the display of SVG color fonts.

13.5.1 SVG Image Files

Again, prepare SVG files outside of Glyphs. One pixel or one point in the SVG will, by default, be scaled to one unit in the font. To keep possible errors and the file size of the resulting fonts at a minimum, we recommend to prepare the SVG files at the right size. Make sure the total size does not exceed the UPM of the font (1000 by default), or whatever other measurement you want the graphics to stay in sync with, e.g., the cap height of your font.

13.5.2 SVG Layers

In the glyphs where you want to add SVG files, add a layer called ‘svg’. Then, drag the SVG file from the Finder into the ‘svg’ layer, or add it via *Glyph > Add Image*. The image will be displayed in its actual scaled size. You can move and scale the SVG image by selecting it, and changing the respective values in the gray Info box (*View > Show Info*, Cmd-Shift-I), or by dragging the handles of the bounding box (*View > Show Bounding Box*, Cmd-Opt-Shift-B). Merely moving the image is possible by simply dragging it with the mouse, or selecting it and using the arrow keys. Add the Shift key for increments of 10 units, and the Cmd key for increments of 100 units.

13.5.3 Using SVG Color Fonts

Once you have one or more glyphs with ‘svg’ layers containing placed SVG files in your Glyphs file, export an OpenType font, preferably as WOFF or WOFF2 for use in Firefox. Glyphs will add the SVG table containing the SVG code of the respective image files. Transformed images will have a modified ‘transform’ attribute in their ‘svg’ tag.

Keep in mind that the in-line display of animated SVGs is very processor-intensive, even if it is only a very simple animation.

14 Error Handling

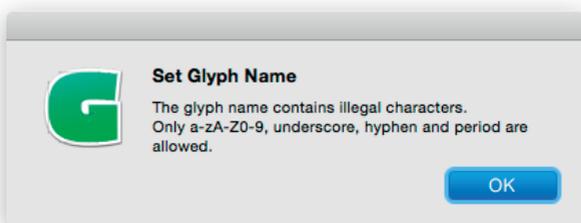
14.1 GLYPH NAMES

The most common source of problems when exporting fonts are bad glyph names. Make sure all your glyph names

- only contain letters A-Z, a-z, numerals 0-9, underscore (`_`), hyphen (`-`) or period (`.`),
- start with a letter (A-Z, a-z), except for `'notdef'`, and non-exporting glyphs, e.g., smart glyphs, that start with an underscore,
- contain no whitespace characters (space, tab, return, etc.), not even at the end,
- and contain no non-ASCII characters (like á, è, ß or ü).

Not adhering to these guidelines may yield an error message at export time like 'There is a problem with a glyph named:', followed by the bad glyph name and a descriptive explanation of the problem within brackets, e.g., 'The glyph name should not contain any space character.' Such error messages can also occur when trying to compile the OpenType feature code in *File > Font Info > Features*.

You can find bad names by searching for space (or other invalid characters) in the search field of the Font view (Cmd-F).



14.2 FONT NAMES

If your error message only entails a POSIX path to a file called 'FontMenuNameDB', set between brackets (e.g., '[/Users/<your username>/Library/Application Support/Glyphs/Temp/<your fontname>/FontMenuNameDB]'), then there probably is an invalid character in the font name or style name. You can use spaces, but no non-ASCII characters.

If you want to use non-ASCII characters in the font name, then you need to assign a *localizedFontName* custom parameter in *File > Font Info > Font* (Cmd-I). For more details, see its entry in the list of custom parameters in the Appendix (p.189). The

same applies to the style name of your font, which can have non-ASCII characters in the parameter *localizedStyleName*.

14.3 DUPLICATE UNICODE VALUES

In some fonts, two or more letters erroneously carry the same Unicode. If you try to import such a font with double encodings, Glyphs will warn you about it. Selecting all glyphs and choosing *Font > Update Glyph Info* will usually fix that problem.

Tip: If you do want to keep a glyph accessible via two different codes, it is better to create a duplicate glyph.

In the reverse situation, where one glyph sports two Unicodes, Glyphs will silently reset the Unicode value based on the glyph name. The usual suspects for this are Delta and Omega. Delta should only be U+0394, but sometimes also has U+2206, the code for the mathematical increment operator (a.k.a. Laplace operator, glyph name: increment). Omega, which should only be encoded with U+03A9, sometimes also sports U+2126, the codepoint for the Ohm symbol (glyph name: Ohm).

14.4 OPENTYPE FEATURE CODE

If all your features are auto-generated, you can usually fix feature code troubles by re-compiling, i.e., clicking the *Update* button in the Feature tab of the Font Info window.

Glyphs tries to pass on any FDK compilation errors in an error dialog. And it also tries to point you to where the problem occurred by reporting the name of the problematic feature and the line number. If it succeeds in doing so, the error dialog will sport a 'Show' button. Click on it and Glyphs takes you directly to the code problem. The following errors may occur:

Contextual substitution clause must have a replacement rule or direct lookup reference. You probably forgot or mistyped the word 'by' in a substitution feature.

DFLT script tag may be used only with the dlft language tag. You tried using a language tag without a preceding script tag, e.g., if you use 'language DEU;', there needs to be a 'script latn;' somewhere before in the feature.

"Feature" statement allowed only in 'aalt' feature. You most likely added your own feature code in the Prefix and forgot to close a feature properly before starting the next one.

Glyph x not in font. You tried referencing a glyph that does not exist or is set to not export. The error message will tell

you which glyph it was looking for in vain. This happens if you mistyped the name of the glyph, accidentally deleted the actual glyph, renamed the glyph, had it removed with the *Remove Glyphs* parameter, or forgot to create the glyph in the first place. Sometimes, the automatically generated features are simply out of date. In this case, recalculating the features by clicking on the *Update* button in the bottom left corner of *File > Font Info > Features* will do.

Unnecessary kerning between two scripts (e.g., a Cyrillic and a Latin letter with each other) is ignored at export time, minimizing the risk for this error.

GPOS feature 'kern' causes overflow of offset to a subtable. The kerning structure is too complicated and causes the glyph positioning table in the font to become too large. An OpenType table must not be larger than 64 kilobytes. Cleaning up and compressing kerning may help: Try the respective functions from the gear menu of the Kerning window (*Window > Kerning*). Also check for unnecessary kerning pairs. Very small values (below 5), and impossible pairs (like Q and Tbar which do not both appear in the same written language) are usually superfluous and can be deleted. Another workaround is to use extended kerning by adding the *Use Extension Kerning* parameter in *File > Font Info > Font*. For more details, see its entry in the list of custom parameters in the Appendix (p. 201).

Invalid token. Most likely one of the class names starts with a whitespace character, or there is a whitespace between the at sign (@) and the class name in the feature code, or there is a whitespace character in a glyph name, or there are invalid non-ASCII characters elsewhere in the feature code. Character restrictions also apply to comments.

Lookup type different from previous rules in this lookup block. You tried to mix contextual and non-contextual rules inside the same lookup, most likely in the 'calt' feature.

MakeOTF error. GPOS feature '...' causes overflow of offset to a subtable. The feature mentioned contains too many positioning rules. If the feature is 'kern', you have too much kerning information stored in your font. Consider setting up classes, compressing kerning, removing some kerning exceptions, or employing the *Use Extension Kerning* custom parameter.

MakeOTF error. GSUB feature '...' causes overflow of offset to a subtable. The feature mentioned contains too many substitution rules. Consider using lookups, removing some substitutions or avoiding the repetition of substitutions in more than one feature.

makeotfGlyphs. The most probable cause is a glyph name containing invalid characters. There usually is some extra information in the dialog which should give you a clue.

makeotfGlyphs [FATAL] line too long. Most likely, a glyph name was too long. The compiler cannot handle glyph names longer than 122 characters.

not in range -32767..32767 (text was "..."). At least one node or anchor in the font is out of bounds. A coordinate value must not exceed $\pm 32,767$. Open the Glyphs file in a text editor and search for the string 'e+'. This way you can find large numbers in exponential notation, e.g., '-9.22337e+18' and see which glyph needs to be fixed. If you do not find anything this way, try the number in the brackets of the error message.

Positioning values are allowed only in the marked glyph sequence, or after the final glyph node when only one glyph node is marked. The syntax of a contextual positioning rule in the 'kern' feature is faulty. At least one glyph name must be marked with a single dumb quote ('), and the number value should come right after the marked glyph, not at the end as in other positioning rules.

Premature end of input. Most likely, one of the glyphs has a bad name. Check if there is a glyph name that contains an equals sign (=), an at sign (@) or brackets ('[' or ']').

Target glyph class in rule doesn't have the same number of elements as the replacement class. You tried substituting a class with a class of a different size. The error dialog will point you to the problematic code line. Make sure the classes are of the same size and in the same order.

Syntax error. This message indicates that the code stored in the OpenType features does not conform to the expected AFDKO syntax. Potential causes include:

- non-ASCII characters in your code,
- one of the feature names does not adhere to the naming rules: exactly 4 lowercase ASCII characters and figures, no whitespace or punctuation,
- mistyped feature commands (like 'sub', 'pos', or 'by'),
- a semicolon missing at the end of a rule (in this case, the error message will contain 'missing ";"'),
- a numerical value missing or mistyped in a positioning lookup (in this case, the error message may also contain 'missing NUM').

14.5 MISSING OUTLINES

Sometimes the font does export, but some glyphs are empty or parts of glyphs are missing.

14.5.1 Open Paths

If the *Remove Overlap* option is active at export, outlines which are not closed are ignored and will not show up in the final font. So, if a glyph appears empty in the OTF, it is a good idea to check if its paths are actually closed.

If you want to keep open paths, because you expand paths into outlines at export time through the help of a custom parameter, then uncheck the *Remove Overlap* option in the *File > Export* (Cmd-E) dialog.

14.5.2 Wrong Path Orientation

Outlines must be oriented counter-clockwise, except for counters, which need to be oriented clockwise. Otherwise, counters may appear 'closed' or missing. You can fix the orientation of selected outlines manually by choosing *Reverse Selected Contours* from their context menu, or rotate all outlines on a layer by invoking the context menu on an empty selection and choosing *Reverse All Contours*. Glyphs will try to apply heuristics to fix outline directions on any number of selected glyphs with *Glyph > Correct Path Direction* (Cmd-Shift-R) or *Glyph > Correct Path Direction for All Masters* (Cmd-Opt-Shift-R).

14.5.3 Multiple Paths on Top of Each Other

Two similar paths on top of each other with varying path orientations may delete each other at export. You can select exactly one whole path by double clicking it. Press the Backspace key to delete it. If it looks like nothing has changed, then you had at least two identical outlines on top of each other.

14.5.4 Outline Incompatibility

Incompatible outlines should not hinder OTF export. Affected glyphs are simply exported empty. If you get a 'Some glyphs are not compatible and will have no outlines' error message, the font will still be exported. You can find more details about fixing incompatible outlines in section 12.4, 'Fix Outline Incompatibility' (p.144).

15 Import and Export

15.1 VECTOR DRAWING APPLICATIONS

15.1.1 Adobe Illustrator

Tip: To quickly get the right scale, draw a rectangle with the height of the capital letters in Glyphs, copy and paste it into an Illustrator artboard and scale the drawings to fit the height of the rectangle.

First, find the right scale for your drawings in Adobe Illustrator. One point in Illustrator corresponds to one unit in Glyphs, i.e., an element that is 100 pt high will end up at a height of 100 units in Glyphs. Alternatively, you can start in Illustrator and set the artboard to a height of your UPM size in points, e.g., 1000 pt if your UPM is 1000. Both ascender and descender should fit inside the artboard. If you set the page origin to the intersection of base line and LSB, the paths will have the right position.

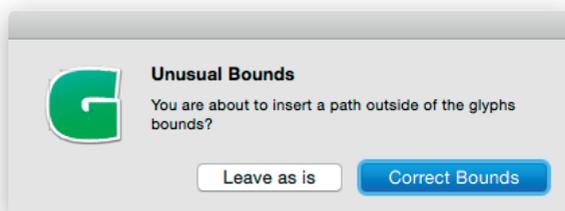
In Illustrator up to Version CS4, set the origin of the page by dragging the cross hair in the top left corner between the rulers. In Illustrator CS5 or later, set the origin of the page in the *Artboard Options*.

15.1.2 Sketch

If you are importing outlines from Sketch, set your contour to 1 pt outline without fill. This will prevent double outlines in Glyphs.

15.1.3 Copy and Paste the Paths

Copy and paste the outlines. A dialog may appear, warning you about inserting something far outside the bounds of the letter. Pushing the *Correct Bounds* button will move the path next to the origin, i.e., the intersection of base line and left sidebearing.



15.2 FONTLAB STUDIO

15.2.1 From FontLab Studio 5 to Glyphs

There is a FontLab macro available for exporting Glyphs files directly out of FontLab Studio. Point your web browser to github.com/schriftgestalt/Glyphs-Scripts/ and download the script called 'Glyphs Export.py'. To install, invoke *Go > Go to Folder* in Finder, enter `~/Library/Application Support/FontLab/Studio 5/Macros/` and place the file in the Macros folder that is displayed then. After restarting FontLab, it will become available in the macro toolbar.

15.2.2 From Glyphs to FontLab Studio 5

Again, there is a Python script to import Glyphs files into FontLab. You can get it from the same Github repository mentioned above. This time, look for a script named 'Glyphs Import.py' and place it in your FontLab Macros folder.

Keep in mind that some features are not available in FontLab, and therefore the result may not be exactly the same. For instance, FontLab Studio cannot nest components, and therefore, the script will decompose them in the conversion.

15.3 UNIFIED FONT OBJECT

15.3.1 Native Saving Format

Glyphs can read and write UFO files natively. When you choose *File > Save As*, you are prompted whether you want to save a Glyphs file or a UFO. Not everything Glyphs can do can be stored in a UFO. If you use UFO as your saving format, Glyphs will warn you if you try to save something that is not supported by UFO.

15.3.2 Exporting to UFO

To be on the safe side, you can keep saving in the Glyphs file format, and export to UFO when you need it by choosing *File > Export > UFO*. If you try to export a Multiple Master file to UFO, the masters will be exported as separate UFO files.

15.3.3 Importing UFO Files

You can simply open a UFO file and save it in the native Glyphs format with *File > Save as*. When importing a font project, Glyphs will try to apply its built-in naming scheme

and sync the metrics of compound glyphs with their base glyphs. To prevent either of the two, you can go to *Glyphs > Preferences > User Settings* and check the options *Keep Glyph Names from Imported Files* and *Disable Automatic Alignment in Imported Files*. To set these options on a per-font basis, go to *File > Font Info > Other Settings*, and check *Use Custom Naming* or *Disable Automatic Alignment*, respectively.

15.4 TYPE1, OPENTYPE, AND TRUETYPE

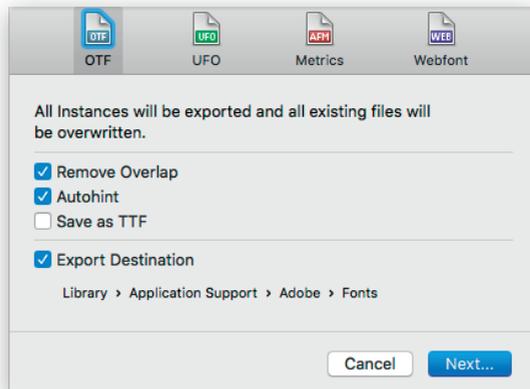
15.4.1 Opening Existing Fonts

While you can open existing PFB, OTF, TTC, and TTF fonts, Glyphs cannot reverse-engineer all the information inside a compiled font file. That means that opening an OTF and exporting it again will produce a file that is different from the original. For instance, you will lose hints and some of the font metadata stored in OpenType tables. It is strongly advised to always work on a copy.

For more details on opening existing fonts, see glyphsapp.com/tutorials/importing-existing-fonts

15.4.2 Generating OpenType/CFF Fonts

Choosing *File > Export* (Cmd-E) will bring up the Export dialog. To generate an OpenType font for use in layout or word processing applications, pick *OTF* as export format. For OpenType/CFF fonts, the *Save as TTF* option must be left unchecked.



The *Remove Overlap* option applies an algorithm to remove path overlaps at export, similar to the *Remove Overlap* filter. In a release font, overlaps must be removed. So, uncheck

this only for testing purposes, to speed up the export. You can safely uncheck this option if you have already removed overlaps in the file or if a custom parameter takes care of the overlap removal.

The *Autohint* option applies the AFDKO autohinting algorithm to all glyphs that do not have any manually set hints. This option expects standard stems and alignment zones to be set correctly (see 10, 'PostScript Hinting', p. 121). Uncheck this option for fonts that should not be hinted, or in order to speed up export for testing purposes.

The *Export Destination* option allows you to set a default location into which the fonts will get exported. If you do not set a destination, Glyphs will bring up a save dialog. Be aware that exporting your font will overwrite any previous instances with the same name in the same export location. This can be useful if you use the Adobe Fonts folder as export destination (see 3.12.7, 'Previewing in Adobe Applications', p. 49).

15.4.3 Generating OpenType/TT Fonts

To export OpenType fonts in their TrueType flavor, choose *File > Export* (Cmd-E) and click on the *OTF* tab of the upcoming dialog. For exporting the font with TrueType outlines, the *Save as TTF* option must be checked.

The *Remove Overlap* and *Export Destination* options work just as with CFF-flavored fonts. See section 15.4.2, 'Generating OpenType/CFF Fonts' (p. 165) for details.

The *Autohint* option applies the *ttfautohint* algorithm by Werner Lemberg to all glyphs. It will override any manually set TrueType hints. You can set options with the custom parameter *TTFAutohint Options*. See its entry in the list of custom parameters in the Appendix (p. 197) for a complete description of what is possible. If, however, you want to employ your own manual hinting (see 11.3, 'Manual Instructions', p. 133), you must keep this option unchecked, or insert the custom parameter *Autohint* with a disabled checkbox in the instances where you want to keep manual TT hints. See the respective entry in the list of custom parameters in the Appendix (p. 180) for further details.

In exported TrueType fonts, compounds are kept in order to reduce file size. Components are decomposed into outlines only if they overlap, e.g., in glyphs such as Ccedilla, Lslash or Tbar. In certain workflows, it can become necessary to keep

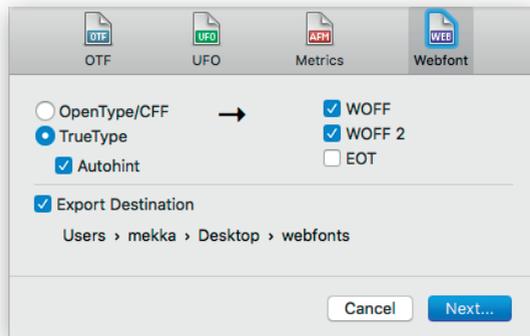
overlaps in these glyphs, while removing overlaps in paths. This facilitates post-processing in other applications. In this case, you can use the custom parameter *Keep Overlapping Components* in the instances. Read more about it in its entry in the list of custom parameters in the Appendix (p.188).

15.5 WEBFONTS

15.5.1 Generating WOFF, WOFF2, EOT

To export OpenType fonts for the web, choose *File > Export* (Cmd-E) and click on the *Webfont* tab of the upcoming dialog. Glyphs can export WOFF and WOFF2 (Web Open Font Format), as well as the TrueType-flavored EOT (Embedded OpenType) for older versions of Internet Explorer.

With the checkboxes on the right, you can export all instances in any or all of the available formats. The selection can be overridden by the custom parameter *Webfont Formats*. See its entry in the list of custom parameters in the Appendix (p.202) for further details.



The radio buttons *OpenType/CFF* and *TrueType* change the OpenType flavor of exported WOFF and WOFF2 fonts. They have no effect on exported EOT files, since they always contain TrueType outlines.

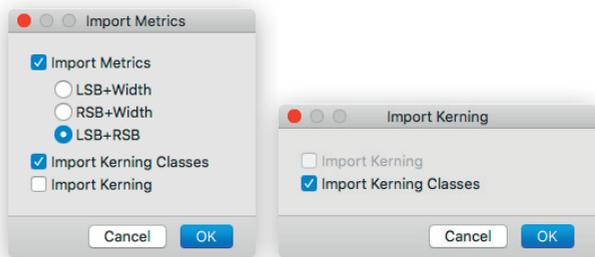
The *Autohint* option triggers PostScript autohinting for CFF-based WOFFs, and *ttfautohint* for TT-based WOFFs and EOTs. This option is overridden by the *Autohint* custom parameter. For more details, see its entry in the list of custom parameters in the Appendix (p.180).

The *Export Destination* setting works like it does for CFF-based OpenType desktop fonts. See section 15.4.2, ‘Generating OpenType/CFF Fonts’ (p.165) for details.

15.6 METRICS

15.6.1 Import Metric Data

You can import kerning data for selected Glyphs via *File > Import > Metrics*, then choosing an existing .glyphs, .metrics, or .ufo file in the upcoming file dialog. In the *Import Kerning* dialog that follows, you can import kerning values and kerning groups (or ‘classes’) from a UFO, and from a Glyphs file, only kerning groups. From a Metrics file, you can also choose to import sidebearings or widths. The dialog adapts depending on the file you selected for import:

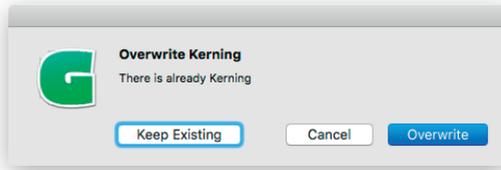


Please note that the glyphs need to be selected before you import kerning groups and values for them. If you want to import the data for the whole font, make sure all glyphs are selected before running the command.

When importing metrics from a .metrics file, Glyphs will disable automatic alignment for all compound glyphs where the metric values deviate from the ones derived from automatic alignment.

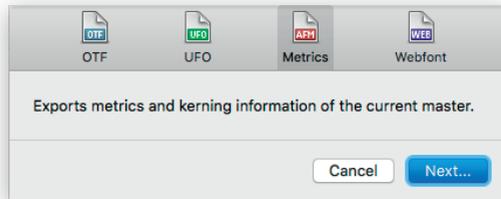
To import kerning values from another Glyphs file, or from one font master to another, open the source file, switch to the master from where you want to import the kerning, then open *Window > Kerning* (Cmd-Opt-K), select the kerning values you want to import, or select all (Cmd-A), and copy them into the clipboard (Cmd-C). Then, switch to the font master you want to import the kerning in (the target), and paste the

kerning data from your clipboard (Cmd-V). Glyphs will warn you if you are overwriting existing kerning.



15.6.2 Export Metrics

If you want to export your sidebearing and kerning values, you can run *File > Export* (Cmd-E) and pick the *Metrics* tab in the dialog sheet that follows. Click on *Next* to choose an export location. In the save file dialog that follows, you can choose between *Metrics File* and *AFM File*.



The *Metrics File* option will create a proprietary file with a *.metrics* suffix, containing all spacing and kerning information of the font. You can then import the *.metrics* file into another *.glyphs* file with *File > Import > Metrics*.

The *AFM File* option will save an old-style Adobe Font Metrics file, which is compatible with other font editors, but due to the constraints of the AFM file format, it cannot contain all the metric information. For instance, AFM does not support group kerning or metric keys.

15.7 PROJECTS

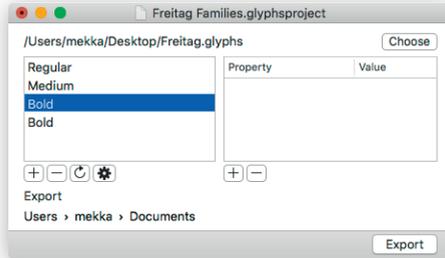
Projects are collections of instance definitions linked to a Glyphs font file, but kept in a separate project file. This is useful for setting up different versions of your fonts without altering or compromising the original *.glyphs* file. Project files carry a *.glyphsproject* suffix.

15.7.1 Setting up a Project

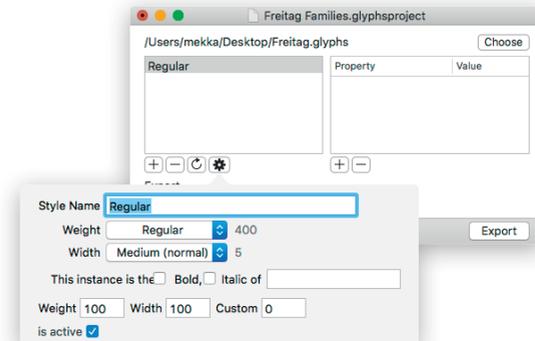
To create a new Glyphs project, select *File > New Project*. Save it, like a regular Glyphs file, with either *File > Save* (Cmd-S) or *File > Save as* (Cmd-Shift-S).

To link the project file to a Glyphs font file, click on the *Choose* button in the top right of the project window. What will be stored in the .glyphsproject file is the file path of the Glyphs file. Once a font file is linked to the project, its instances will be imported in the instances sidebar on the left.

Note that the font file does not need to be opened in Glyphs for linking it to the project. It only needs to be saved on disk.



You can rearrange instances by dragging and dropping. You can duplicate existing instances by selecting them and dragging them to a new location in the list with the Option key held down. Alternatively, you can copy (Cmd-C) and paste them (Cmd-V). You can add a new instance with the plus button in the lower left, and delete selected instances with the minus button next to it. You can reset the list of instances to the setup stored in the .glyphs file with the refresh button. You can edit the style name of the instance, as well as settings for weight, width, style linking, design space coordinates, and the *Is Active* export toggle through the gear menu.



You can manage custom parameters of the currently selected instance in the parameter list on the right. It works like the Custom Parameters fields in the *Instances* tab of *File > Font Info* (Cmd-I). See section 7.3.6, 'Custom Parameters' (p. 93) for further details. For a list of available custom parameters, see section 17.3, 'Custom Parameters' (p. 180).

15.8 EXPORTING A PROJECT

To export all font instances of the project, first make sure you have saved the `.glyphsproject` file, and that the path to the `.glyphs` file is valid. Then, pick an export path by clicking on the displayed path in the *Export* section at the bottom of the project window. Once the export path is set, you can click on the *Export* button in the bottom right corner of the window to initiate the creation of all active font instances.

Note that also for exporting,
the original `.glyphs` file
does not need to be open
in the application.

16 Extensions

16.1 SCRIPTS

Python scripts are text files with a .py ending containing code in the Python programming language. They can be used to automate tasks in Glyphs. You can make Python scripts accessible from inside the application if you put them into the Scripts folder inside the Application Support folder of Glyphs. The quickest way to get there is to choose *Script > Open Scripts Folder* (Cmd-Shift-Y). You can arrange your scripts in subfolders. After you have placed .py files there, you can hold down the Option key and choose *Script > Reload Scripts* (Cmd-Opt-Shift-Y). The scripts should then show up in the *Scripts* menu. Python scripts for Glyphs should begin with a comment that stipulates the script title that shows up in the pull-down menu, e.g.:

```
#MenuItem: Rotate Glyphs
```

Whatever is written into the `__doc__` string will be shown as the script's tooltip when the mouse hovers over the script's menu entry in the *Script* menu:

```
__doc__="Create effect for selected glyphs."
```

There is extensive documentation of the Python Scripting API available on the Glyphs website:

- docu.glyphsapp.com

For beginners, we provide a multipart introductory tutorial in the Tutorials section of the Glyphs website:

- glyphsapp.com/tutorials/articles/tag:scripting

And you can find and study many third-party scripts for Glyphs on GitHub. You will find links to some select repositories on the Glyphs website as well:

- glyphsapp.com/extend

16.2 PLUGINS

16.2.1 Manual Plugin Installation

You install a plugin by opening it with Glyphs, e.g., by double clicking it, or dragging it onto the app icon. The plugins are then automatically moved into the Plugins folder inside the Application Support folder of Glyphs. Alternatively, you can move the plugin or an alias of the file there manually. You can uninstall it again by moving it out of that folder. Quick ways of accessing the Plugins folder are (1) choosing *Script > Open*

Scripts Folder and then clicking on the *Plugins* folder next to the *Scripts* folder, or (2) opening *Glyphs > Preferences > Addons > Plugins*, right-clicking on one of the installed *Plugins* and choosing *Show in Finder* from the context menu.

There are various kinds of plugins that can be installed in *Glyphs*. They can be differentiated with their file name suffix:

- *Reporter* plugins (file name suffix `.glyphsReporter`) extend the *View* menu with an additional *Show* option, and draw on the canvas.
- *Filter* plugins (`.glyphsFilter`) process the selected glyphs. Some have dialogs for options, some do not. Some can be triggered by an instance custom parameter at export time. Refer to the plugin documentation for details.
- *File Format* plugins (`.glyphsFileFormat`) for an additional format for opening or exporting.
- *Palettes* (`.glyphsPalette`) provide extra functionality in the *Palette* sidebar (*Window > Palette*, `Cmd-Opt-P`).
- *Select Tool* plugins (`.glyphsTool`) will show up as new tools in the toolbar.
- *General* plugins (`.glyphsPlugin`) for other functionality, not covered by the categories listed above.

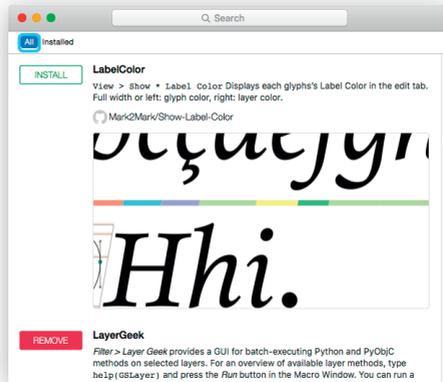
You should be able to find some *Glyphs* plugins via the *Extend* section of the *Glyphs* website:

- glyphsapp.com/extend

16.2.2 Plugin Manager

You can comfortably install a selection of freely available plugins through *Window > Plugin Manager*. In the window that appears, you can search for a plugin with the search box at the top, read the descriptions, or click the provided links to

visit the website for the respective plugin. If the description contains an image, you can click it to preview it at full size.



Since plugins are loaded at startup time, you will need to restart Glyphs for changes to take effect after installation or uninstallation.

To install a plugin, click the green *Install* button next to the plugin name. To uninstall, click the red *Remove* button. You can click the *Installed* button at the top of the window to show only installed plugins, for easier uninstallation. Click *All* to show all, installed and uninstalled plugins again.

If you have written a plugin for Glyphs and want it to appear in the Plugin Manager, you can make a pull request on the plugin list. Details are described in the readme files of the GlyphsSDK (see below).

16.3 SDK

The GlyphsSDK is an open-sourced software development kit (SDK) for Glyphs. Its purpose is to facilitate software development for the Glyphs ecosystem, be it for plugins, scripts, or for a standalone app that processes Glyphs data. The SDK contains the documentation for the scripting API, a description of the .glyphs file format, and templates for plugin development, for both ObjectiveC in Xcode and Python via the PyObjC bridge. Download the GlyphsSDK from GitHub at:

- github.com/schriftgestalt/GlyphsSDK

17 Appendix

17.1 AUTOMATIC FEATURE GENERATION

Glyphs can automatically generate a number of OpenType features if it finds glyphs with certain names in the font.

aalt	<i>All Alternatives</i>	Glyphs automatically builds the aalt feature based on all features that substitute glyphs.
liga	<i>Ligature</i>	Join the glyph names of the components with an underscore ('_'). Some common ligatures (f_f_i, f_f_l, f_f, fi, fl, lu_lakkhangyao-thai, ru_lakkhangyao-thai) are automatically placed in the liga feature, all others go into dlig by default. However, if you want to force a ligature into the liga feature, you can add a 'liga' suffix to its name, e.g., f_b.liga or yi_yi-cy.liga.
dlig	<i>Discretionary Ligatures</i>	Join the glyph names of the components with an underscore ('_'), e.g., f_odieresis.
hlig	<i>Historical Ligatures</i>	Ligatures with longs, or with a .hlig or .hist suffix.
rlig	<i>Required Ligatures</i>	Add 'rlig' to the ligature name. Also triggered by lam_alef-ar, lam_alefHamzaabove-ar, lam_alefHamzabelow-ar, lam_alefMadda-ar, lam_alef-ar.fina, lam_alefHamzaabove-ar.fina, lam_alefHamzabelow-ar.fina, lam_alefMadda-ar.fina, lam_alefWasla-ar, lam_alefWasla-ar.fina
c2sc	<i>Small Capitals from Capitals</i>	Add 'sc', 'c2sc' or 'smcp' to the glyph name.
smcp	<i>Small Capitals</i>	Add 'sc', 'c2sc' or 'smcp' to the glyph name.
c2pc	<i>Petite Capitals from Capitals</i>	Add 'pc', 'c2pc' or 'pcap' to the glyph name.
pcap	<i>Petite Capitals</i>	Add 'pc', 'c2pc' or 'pcap' to the glyph name.
sup	<i>Superscript</i>	Add 'sup' to the glyph name. Or extend figure names with 'superior', without the period, e.g., 'onesuperior'.
sub	<i>Subscript</i>	Add 'inferior' or 'sub' to the glyph name.
sinf	<i>Scientific Inferiors</i>	Add 'sinf' or 'sub' to the glyph name.

If you have separate sets for c2sc and smcp, you can use 'c2sc' for uppercase glyph names and 'smcp' for lowercase glyph names.

If your font does not differentiate between subscript and scientific inferior, simply use one set of 'sub' glyphs and Glyphs will create both features with it.

afrc	<i>Alternative Fractions</i>	Add figures, slash, and any of these nut fraction glyphs to your font: oneovertwo, zerooverthree, oneovertthree, twoovertthree, oneoverfour, threeoverfour, oneoverfive, twooverfive, threeoverfive, fouroverfive, oneoversix, fiveoversix, oneoverseven, twooverseven, threeoverseven, fouroverseven, fiveoverseven, sixoverseven, oneovereight, threeovereight, fiveovereight, sevenovereight, oneovernine, twoovernine, fourovernine, fiveovernine, sevenovernine, eightovernine, oneoverten, threeoverten, sevenoverten, nineoverten, oneovereleven, twoovereleven, threeovereleven, fourovereleven, fiveovereleven, sixovereleven, sevenovereleven, eightovereleven, nineovereleven, tenovereleven, oneovertwelve, fiveovertwelve, sevenovertwelve, elevenovertwelve, oneoveronehundred.
frac	<i>Fractions</i>	The frac feature is generated from the .numr, .dnom and fraction glyphs. If they are not present in the font, then the feature will be composed from any pre-built fractions available in the font, like onehalf, onequarter, threequarters etc.
dnom	<i>Denominators</i>	Add 'dnom' to the glyph name.
numr	<i>Numerators</i>	Add 'numr' to the glyph name.
onum	<i>Oldstyle Figures</i>	Add 'osf' (for proportional oldstyle figures) or 'tosf' (for tabular oldstyle figures) to the glyph name. Can be applied to other characters as well, e.g., currency signs.
tnum	<i>Tabular Figures</i>	Add 'tf' (for tabular figures) or 'tosf' (for tabular oldstyle figures) to the glyph name. Can be applied to other characters as well, e.g., currency signs.
pnum	<i>Proportional Figures</i>	Add 'osf' (for proportional oldstyle figures) or 'lf' (for proportional lining figures) to the glyph name. Can be applied to other characters as well, e.g., currency signs.
lnum	<i>Lining Figures</i>	Add 'lf' (for proportional lining figures) or 'tf' (for tabular figures) to the glyph name. Can be applied to other characters as well, e.g., currency signs.

Tip: do not use the figure suffix which would apply to your default figures, e.g., if your default figures are proportional oldstyle figures, do not use figures with an .osf suffix.

ordn	<i>Ordinals</i>	Automatically built if default figures, numero, ordfeminine and ordmasculine are found in the font.
mgrk	<i>Mathematical Greek</i>	Next to Delta, Omega, Pi, Sigma, alpha, mu, phi, epsilonLunateSymbol, epsilonLunateReversedSymbol, also add: increment, Ohm, product, summation, proportional, micro, phiSymbol, element, containsasmemberSmall. Since the feature is considered deprecated, Glyphs will not automatically add it, but only automatically update it if you have added it manually.
ornm	<i>Ornaments</i>	Add 'ornm' to the glyph name of letters A–Z or a–z. Also make sure you have the glyph 'bullet' in your font.
hist	<i>Historical Forms</i>	Add 'hist' to the glyph name.
case	<i>Uppercase Forms</i>	Add 'case' to the glyph name or 'lf' to the name of a figure.
cpSP	<i>Capital Spacing</i>	Add uppercase letters to your font, and choose <i>Capital Spacing</i> from the plus menu in the bottom left corner of <i>File > Font Info > Features</i> (Cmd-I).
locl	<i>Localized Forms</i>	Add 'loclXXX' to the glyph name, where XXX represents the three letter language tag, e.g., 'loclENG' for English, or 'loclSVE' for Swedish. There also is a built-in list of glyphs which which trigger localizations: <ul style="list-style-type: none"> • idotaccent, i.TRK or i.loclTRK: trigger i substitutions for TRK, AZE, CRT, KAZ and TAT if i is present. • Scommaaccent, Tcommaaccent, scommaaccent, and tcommaaccent: trigger substitutions for ROM and MOL if Scedilla, Tcedilla, scedilla, and tcedilla are present. • periodcentered.loclCAT (add a .case suffix for uppercase; L_periodcentered_L.loclCAT, l_periodcentered_l.loclCAT; or the legacy Ldot, ldot: trigger ella geminada (punt volat) substitutions for CAT if L, l, and periodcentered are also present in the font. • Iacute_J.loclNLD and iacute_j.loclNLD, or Iacute and jacute: trigger accented ij substitutions for NLD if Iacute, iacute, J, and j are present. • six-ar and numbers with .urdu suffix (e.g. four-persian.urdu) will trigger URD localization for Persian.

For a complete list of language tags, see: microsoft.com/typography/otspec/language-tags.htm

Find out more about these languages and other localizations: glyphsapp.com/tutorials/tag/languages

cv01– cv99	<i>Character Variants</i>	Add ‘cv01’ through ‘cv99’ to the glyph name.
ss01– ss20	<i>Stylistic Set</i>	Add ‘ss01’ through ‘ss20’ to the glyph name. If you add ‘Name:’ plus the feature’s descriptive name to the feature note at the bottom, Glyphs will generate the feature name entries for stylistic set names. Alternatively, you can add the complete featureNames AFDKO code to the note (see 7.4.3, ‘Manual Feature Code’, p. 94).
salt	<i>Stylistic Alternates</i>	By default, Glyphs will duplicate the ss01 feature in salt. Adobe Illustrator and Adobe Photoshop make use of this feature in their OpenType palettes.
swsh	<i>Swashes</i>	Add ‘swsh’ to the glyph name.
titl	<i>Titling</i>	Add ‘titl’ to the glyph name.
init	<i>Initial Forms</i>	Add ‘init’ to the glyph name.
medi	<i>Medial Forms</i>	Add ‘medi’ to the glyph name.
med2	<i>Medial Forms</i>	Add ‘med2’ to the glyph name. Used only with the Syriac script.
fina	<i>Terminal Forms</i>	Add ‘fina’ to the glyph name.
fin2	<i>Terminal Forms</i>	Add ‘fin2’ to the glyph name. Used only with the Syriac script.
fin3	<i>Terminal Forms</i>	Add ‘fin3’ to the glyph name. Used only with the Syriac script.
hwid	<i>Half Widths</i>	Add ‘half’ to the glyph name.
vrt2	<i>Vertical Alternates and Rotation</i>	Add ‘vert’ to the glyph name.
akhn	<i>Akhands</i>	Triggered by k-deva, j-deva, ssa-deva, nya-deva and k_ssa-deva, j_nya-deva.
blwf	<i>Below Base Forms</i>	Triggered by ra-deva, halant-deva and rashtrasign-deva.
cjct	<i>Conjunct Forms</i>	Triggered by conjunct clusters in Devanagari and other Indic scripts.
half	<i>Half Forms</i>	Triggered by half-form glyphs ending in ‘Halfform’ in conjunction with halant in Devanagari and other Indic scripts.
nukt	<i>Nukta Forms</i>	Triggered by nukta ligatures ending in ‘Nukta’ and the script abbreviation, in conjunction with the same glyphs without nukta, in Devanagari and other Indic scripts.

rkrf	<i>Rakar Forms</i>	Triggered by rakar ligatures in conjunction with the isolated glyphs and halant, in Devanagari and other Indic scripts.
rphf	<i>Reph Forms</i>	Triggered by ra-deva, halant-deva and reph-deva, or an analogous glyph structure in other Indic scripts
ccmp	<i>Glyph Composition and Decomposition</i>	A wide range of glyph constellations in various scripts will trigger automatic creation of ccmp. E.g., idotless and jdotless next to i, j and combining top marks, will trigger a ccmp lookup for Latin, which replaces the dotted with the dotless glyphs before top marks.
mark	<i>Mark to Base Positioning</i>	Add anchors without underscores to your base letters, like ‘top’ or ‘bottom’. Then, add combining marks (e.g., acutecomb) with underscore anchors (e.g., ‘_top’ or ‘_bottom’ to your font. Their width is automatically set to zero at export. This GPOS feature is not added to the <i>Features</i> tab, but calculated at export.
mkmk	<i>Mark to Mark Positioning</i>	Add combining marks (U+0300 and above) with both underscore (e.g., ‘_top’) and non-underscore anchors (e.g., ‘top’) to your font. This GPOS feature is not added to the <i>Features</i> tab, but calculated at export.

17.2 AUTOMATIC CLASS GENERATION

Some OpenType classes can also be automatically generated, or updated if present in the Font.

All	All glyphs in the font. Must be manually added, will be updated automatically if its <i>Generate Feature Automatically</i> option is checked.
AllLetters	All glyphs of category Letter present in the font. Must be manually added, but can be updated automatically.
ArabicLetters	All glyphs of category ‘Letter’ and of the Arabic script present in the font.
DevaHalfforms	All Devanagari half forms present in the font.

Uppercase All glyphs of category ‘Letter’ and subcategory ‘Uppercase’. Generated when uppercase letters and the csp feature are in the font.

17.3 CUSTOM PARAMETERS

All quotes in this chapter are taken from the Microsoft OpenType specification at microsoft.com/typography/otspec/ unless stated otherwise.

Custom parameters have a property and a value. In this appendix, the properties are printed in bold. The short description next to it explains the respective values and the function of the parameter.

In the *Custom Parameters* field of the *Font, Masters and Instances* tabs of *File > Font Info* (Cmd-I), enabled parameters are displayed in black, disabled parameters are displayed in gray. You can quickly disable a parameter by changing its property name, e.g., adding an x to the beginning.

Custom parameters in camelcase are defined in the UFO specification, and change a font information, while capitalized ones are specific to Glyphs and usually change something in the font, e.g., run a filter on the outlines. UFO parameters follow the naming convention for Font Info properties as set forth in the UFO 3 specification published in March 2012. Glyphs also makes use of a simplified naming convention. Wherever possible, you can leave out the prefix of the keyword, e.g., instead of *openTypeNameDescription*, you can simply use *description*, or *blueScale* instead of *postscriptBlueScale*. Both long and short versions work side by side, though.

UFO3 Font Info properties are described on unifiedfontobject.org/versions/ufo3/fontinfo.html

Autohint *boolean* Forces autohinting for the given instance, regardless of the setting in the Export dialog

blueScale *float* BlueScale value. This corresponds to the Type 1/CFE BlueScale field. Controls the font size until which overshoot display is suppressed. Calculated as (pointsize at 300 dpi – 0.49) ÷ 240, e.g., 0.039625 for 10 points at 300 dpi. If you do not set the value yourself, blueScale defaults to 0.037, which corresponds to 9.37 points at 300 dpi or 39 pixels per em. This means that, in this case, overshoots will be visible if at least 40 pixels are used to display an em. The maximum blueScale value depends on the sizes of your alignment zones. The maximum pointsize at 300 dpi is calculated as $0.49 + 240 \div \text{largest alignment zone size}$, which corresponds to a PPM (size in pixels per em) of $2.04 + 1000 \div \text{largest alignment zone size}$. The product of (maximum pointsize – 0.49) × (largest alignment zone height) must be less than 240.

For example, your largest zone is 21 units deep, thus: $2.04 + 1000 \div 21 = 49.659$, so the maximum PPM at which overshoots can be suppressed is 49. The corresponding maximum pointsize is $0.49 + 240 \div 21 = 11.919$ points at 300 dpi, thus the blueScale value cannot exceed $(11.919 - 0.49) \div 240 = 0.04762$.

blueShift *integer or float* BlueShift value. This corresponds to the Type 1/CFF BlueShift field. Default value is 7. Extends for very small glyph features beyond the font size indicated by blueScale. Overshoots inside an alignment zone are displayed if: (a) they are equal to or larger than BlueShift and (b) if they are smaller than BlueShift but larger than half a pixel. E.g. blueScale is set to suppress overshoots until 32 PPM, blueShift is 6, overshoots are 12 units deep. The stroke endings are slightly slanted and extend 5 units below the baseline. Between 0 and 32 PPM, the baseline will be kept completely level. Starting at 33 PPM, the overshoots will kick in. But the stroke endings will stay flat, because 5 units do not cover half a pixel until 100 PPM.

CJK Grid *integer* Number of rows and columns of a dotted-line grid displayed when editing CJK glyphs. You can set number of rows and columns separately with the *CJK Grid Horizontal* and *CJK Grid Vertical* parameters. No grid is displayed when none of these parameters are set. This parameter can be localized like the CJK Guide parameter.

CJK Grid Horizontal *integer* Number of columns of a dotted-line grid displayed when editing CJK glyphs. This parameter can be localized like the CJK Guide parameter.

CJK Grid Vertical *integer* Number of rows of a dotted-line grid displayed when editing CJK glyphs. This parameter can be localized like the CJK Guide parameter.

CJK Guide *float or string* Percentage of inset for CJK guide squares, e.g., 10 for 10 percent. If set, Glyphs will display a second square guide for the virtual body in CJK glyphs. You can localize the parameter by preceding the value with the script name, e.g., 'kana:5'. If you want to define virtual bodies for more than one script, add more CJK Guide parameters.

codePageRanges *list* Sets the appropriate bits of the ulCodePageRange1 and ulCodePageRange2 entries in the OS/2 table. 'This field is used to specify the code pages encompassed by the font file in the cmap subtable for platform 3, encoding ID 1 (Microsoft platform).' Every activated 'code page is considered

functional. Each of the bits is treated as an independent flag and the bits can be set in any combination. 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.’

Color Palette *color palette* Exports a CPAL OpenType table for Microsoft-style color fonts (see 13.3, ‘Microsoft Color Fonts’, p. 153). This parameter allows Glyphs to display a preview of color glyphs in Font view.

compatibleFullName *string* Compatible full name (Mac only). Corresponds to the OpenType name table name ID 18. If not set, the value for name table ID 18 is calculated from Family Name plus space plus Style Name of the respective instance. ‘On the Macintosh, the menu name is constructed using the FOND resource. This usually matches the Full Name. If you want the name of the font to appear differently than the Full Name, you can insert the Compatible Full Name in ID 18.’

Compatible Name Table *boolean* Exports a legacy name table as expected by some Mac apps (e.g., Quark XPress, FontExplorer).

copyright *string* Copyright statement. Overrides the entry in the Copyright field in the Font tab of the Font Info. Corresponds to the OpenType name table name ID 0.

Decompose glyphs *list* Decomposes the compound glyphs listed. This can be useful if you want to avoid changing of compounds when one of the components is being changed with the *Rename Glyphs* parameter.

description *string* Description of the font. Corresponds to the OpenType name table name ID 10: ‘description of the typeface. Can contain revision information, usage recommendations, history, features, etc.’

DisableAllAutomaticBehaviour *boolean* If checked, all automatisms at export are switched off. This includes the zeroing of nonspacing glyph widths, automatic recalculation of OpenType features, and the renaming of glyphs into their production names. Makes most sense at the Font level. Should only be employed if very specific production requirements dictate its use.

Disable autohinting for glyphs *list* Excludes listed glyphs from the PostScript autohinting at export time. (TT autohinting cannot be disabled on a per-glyph basis.) This can be useful if some glyphs do not lend themselves for hinting, e.g., ornaments.

- Disable Last Change** *boolean* Prevents the Last Changed Date from being written into the .glyphs file. This can facilitate version control.
- Disable Masters** *list* Disables all masters with the specified names. Intended mainly for specific production workflows, or for testing purposes, to see if interpolation still behaves as expected if you leave out one of the intermediate masters.
- Disable Subroutines** *boolean* If set, CFF outline subroutinization is disabled when the font is exported. Use this (a) when the font has complex outlines with many nodes and does not export at all, or (b) for testing purposes when the font has many glyphs, e.g., a CJK font, and takes too long to compile every time you export.
- Don't use Production Names** *boolean* If checked, Glyphs will not automatically rename glyphs of the final font file according to the internal glyph database, but export the current glyph names. Some applications and systems, amongst which the text engine of OS X 10.4, expect the AGL naming scheme, though. This is equivalent to the *File > Font Info > Other Settings > Use Custom Naming* setting, and intended for users who want to employ their own custom naming scheme.
- EditView Line Height** *integer* Sets the line height for text set in an Edit tab. Useful if you have unusual vertical metrics, and the default leading seems inappropriate. Has no effect on the exported font file.
- Export Glyphs** *list* Exports all glyphs listed, regardless of whether the glyph was set to export or not.
- Export Path** *string* A POSIX-style path to a folder into which the files are going to be exported.
- Family Alignment Zones** *list* This parameter can help create a more consistent screen appearance at low resolutions, even if the overshoots differ in the individual weights. It is a good idea to reduplicate the alignment zones of the most important font in your family, usually of the Regular or Book instance. A rasterizer will then try to align all weights if the height difference between the individual weight and the family alignment is less than one pixel. Important: For this mechanism to work, family alignment zones must be com-

Above: without family alignment.

Below: with family alignment.

patible with the alignment zones set up in the masters.

family alignment.

family alignment.

familyName *string* Family name. Overrides the entry in the Family Name field in the Font section of the Font Info (see 7.1.1, 'Family Name', p. 84). Corresponds to the OpenType name table name IDs 1 and 4. Used to calculate IDs 3, 4 and 6.

fileName *string* Name for the font file, without the dot suffix, i.e., without '.otf', etc. Gives you the chance to export two versions of the same font style name without the second file overwriting the first one.

Filter *string* Triggers Glyphs filters or app functions in an instance, after decomposition of compound glyphs. The values for the default filters are as follows:

- AddExtremes
- HatchOutlineFilter; OriginX:<x>; OriginY:<y>; StepWidth:<distance>; Angle:<angle>; Offset:<thickness>
- OffsetCurve; <x>; <y>; <make stroke>; <position/auto>
- RemoveOverlap
- Roughenizer; <segment length>; <x>; <y>; <angle>
- RoundCorner; <radius>; <visual correction>
- RoundedFont; <stem>
- Transformations; LSB:<±*/shift>; RSB:<±*/shift>; Width:<±shift>; ScaleX:<percent>; ScaleY:<percent>; Slant:<amount>; SlantCorrection:<bool>; OffsetX:<amount>; OffsetY:<amount>; Origin:<select>

The boolean values (<make stroke>, <visual correction>, <bool>) are 1 for yes and 0 for no. The value for <position/auto> must be a floating point number where 0.0 represents 0%, and 1.0 stands for 100%, or the string 'auto' for *Auto Stroke*. The <stem> value in the RoundedFont parameter is optional. The <select> value in the Transformations parameter can be a number from 0 to 4, representing the five options in *Filter > Transformations > Transform > Origin*: cap height (0), half cap height (1), x-height (2), half x-height (3), baseline (4). If you want a filter to be applied only to some glyphs, add 'include:' or 'exclude:', followed by space- or comma-separated glyph names, e.g., 'RemoveOverlap; exclude:a,b,c'.

Tip: When applying the Round Corner parameter, use negative values for <radius> to round inside (i.e., white) corners.

If you are using third-party filters, refer to their documentation for the parameter string. In particular, the include and exclude options may not be available.

If you want to apply filters before decomposition, use these values with the *PreFilter* property, see its entry in the list of custom parameters in the Appendix (p.193).

fsType *list* A list of bit numbers indicating the embedding type. The bit numbers are listed in the OpenType OS/2 specification. Corresponds to the OpenType OS/2 table fsType field. 'Type flags. Indicates font embedding licensing rights for the font. 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 vendor of the font.

The OpenType Font Embedding DLL Specification and DLL release notes describe the APIs used to implement support for OpenType font embedding and loading. Applications that implement support for font embedding, either through use of the Font Embedding DLL or through other means, must not embed fonts which are not licensed to permit embedding. Further, applications loading embedded fonts for temporary use (see Preview & Print and Editable embedding below) must delete the fonts when the document containing the embedded font is closed.' You can set fsType to one of these five states:

- *Not set*: '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.'
- *Forbidden*: 'Restricted License embedding: Fonts that have only this bit set must not be modified, embedded or exchanged in any manner without first obtaining permission of the legal owner. Caution: For Restricted License embedding to take effect, it must be the only level of embedding selected.'
- *Editable*: 'When this bit is set, the font may be embedded but must only be installed temporarily on other systems. In

Please note that the embedding type is really just a usage suggestion for an application, not an actual protection mechanism. An application may ignore the fsType setting.

contrast to Preview & Print fonts, documents containing Editable fonts may be opened for reading, editing is permitted, and changes may be saved.'

- *Preview & Print*: 'When this bit is set, the font may be embedded, and temporarily loaded on the remote system. Documents containing Preview & Print fonts must be opened "read-only;" no edits can be applied to the document.'
- *Subsetting forbidden*: 'When this bit is set, the font may not be subsetted prior to embedding. Other embedding restrictions also apply.'

GASP Table settings Sets the gasp table ('grid-fitting and scan-conversion procedure') for TrueType fonts. It controls the two PPM thresholds at which the recommended on-screen rendering behavior changes. The gasp table contains rendering recommendations for both a traditional grayscale and a ClearType subpixel renderer. However, keep in mind that a renderer may ignore the data stored herein. 'This table contains information which describes the preferred rasterization techniques for the typeface when it is rendered on grayscale-capable devices. This table 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.' The default threshold sizes are 8 and 20 PPM. Because there are two thresholds, three ranges can be differentiated:

- *no hinting & symmetric*: Until the first threshold size, no gridfitting is applied, and text is rendered with antialiasing wherever possible. 'At very small sizes, the best appearance on grayscale devices can usually be achieved by rendering the glyphs in grayscale without using hints.'
- *hinting & asymmetric*: Between the two threshold sizes, the renderer is recommended to apply gridfitting and suppress grayscale. 'At intermediate sizes, hinting and monochrome rendering will usually produce the best appearance.' In ClearType, the matter is handled asymmetrically, i.e., vertical gridfitting is applied, while horizontally, subpixel rendering is used.
- *hinting & symmetric*: Beyond the thresholds, the rasterizer is instructed to apply gridfitting and render the shapes in grayscale. 'At large sizes, the combination of hinting and grayscale rendering will typically produce the best appearance.' The ClearType rasterizer is instructed to apply symmetric smoothing, i.e., to use anti-aliasing in y direction in

addition to horizontal subpixel rendering. 'At display sizes on screen, [...] this new improvement of the Windows font renderer produces smoother and cleaner-looking type' (Now Read this: The Microsoft ClearType Font Collection, Microsoft 2004, p. 14).

glyphOrder *string* Sets the order of glyphs in both the working file and the final font. Glyph names need to be separated by newlines. You can copy and paste the content of a List Filter. Glyphs not listed but still in the font will be appended after listed glyphs, in the default order that Glyphs employs.

Grid Spacing *float* Set the coordinate precision for the resulting font, in font units. The value corresponds to the quotient of the *Grid Spacing* value divided by the *Subdivision* value in *File > Font Info > Other Settings* (see 7.5.1, 'Grid Spacing and Subdivision', p. 96). Use 0.0 for maximum precision in interpolated or expanded instances, or 0.01 for a grid with a coordinate precision a hundred times finer than the default unit. Its main purpose is to avoid the rounding of point coordinates in very thin interpolations.

Has WWS Names *boolean* Sets bit 8 of the fsSelection entry in the OS/2 table: According to the OpenType specification, this bit indicates that 'the font has "name" table strings consistent with a weight/width/slope family without requiring use of "name" IDs 21 and 22.' This makes sense only if the naming of your font already adheres to the WWS scheme.

hheaAscender *integer* Height of the ascender as stored in the hhea (horizontal header) table. 'Typographic ascent (distance from baseline of highest ascender).'

hheaDescender *integer* Depth of the descender as stored in the hhea table, represented as a negative number. 'Typographic descent (distance from baseline of lowest descender).'

hheaLineGap *integer* The recommended interlinear whitespace as stored in the hhea table. 'Typographic line gap.'

Instance Preview *list* Changes the preview string of an instance in *File > Font Info > Instances* from the default 'Aang126' to the glyph names listed. Especially useful if you are designing a non-Latin typeface.

InterpolationWeightY *integer* Vertical interpolation value. In an instance, you can differentiate between interpolation along the x axis and interpolation along the y axis by introducing this custom parameter. For it to take effect, it must differ from the interpolation weight of the instance. Be careful, as

this can lead to deformation in diagonals. We advice to keep the `InterpolationWeightY` close to the normal `Weight` interpolation value.

E.g., there are two masters at weight 20 and 100, and an instance with a weight interpolation value of 50. But the horizontals look too thin. They would look right at 60, but then the verticals are too thick. So, you keep your instance at 50, but add the custom parameter *InterpolationWeightY* with a value of 60. Now, the vertical stems (x coordinates) are still interpolated with 50, and the horizontals (y coordinates) with 60.

isFixedPitch *boolean* Sets the `isFixedPitch` flag in the post table. Indicates whether the font is monospaced. Software can use this information to make sure that all glyphs are rendered with the same amount of pixels horizontally at any given PPM size.

italicAngle *integer or float* Italic angle. This must be an angle in clockwise degrees from the vertical. Overrides the entry in the `Italic Angle` field in the Masters tab of the Font Info. Useful for upright fonts with an angle other than 0°, because OSX interpretes non-zero angles as italic. Affects the `CFF ItalicAngle`, the post `italicAngle`, the x offsets of the OS/2 subscript and superscript values, as well as the `hhea caretSlopeRise` and `caretSlopeRun` entries.

In the CFF and post tables, the italic angle is stored as counter-clockwise from the vertical. Glyphs will convert your value accordingly.

Keep Glyphs *list* List of glyphs that will be kept in the exported font. All other glyphs will be discarded, and kerning and automatic feature code will be updated accordingly. Works as the opposite of the *Remove Glyphs* parameter, useful for webfont subsetting in order to achieve smaller file sizes. *Remove Glyphs* and *Keep Glyphs* are mutually exclusive.

Keep Overlapping Components *boolean* Does not decompose compound glyphs with overlapping components, such as in `Ccedilla`. Useful for post-production of TrueType fonts.

license *string* License description. Corresponds to the OpenType name table name ID 13, the ‘description of how the font may be legally used, or different example scenarios for licensed use. This field should be written in plain language, not legalese.’

licenseURL *string* URL for the license. Corresponds to the OpenType name table name ID 14. ‘URL where additional licensing information can be found.’ Make sure it starts with the protocol specification, typically ‘`http://`’.

Link Metrics With First Master *boolean* If checked, keeps the sidebearings and the kerning of all masters and color layers in sync with the first master. In effect, you only have to space and kern the first master. This is especially useful for color fonts or fonts that should not change their metrics throughout their weights.

Local Interpolation *string* Apply different interpolation values for specified glyphs. The string must be of the format '<weight>; <width>; <custom>; include: <comma-separated glyph names>', i.e., start with a list of semicolon-separated interpolation values, followed by another semicolon, the string 'include:', followed by a comma-separated list of glyph names. For a single-axis setup, a single interpolation value suffices. E.g., '120; include: a, g, s' uses Weight interpolation value 120 just for the glyphs a, g and s, while all other glyphs are interpolated according to the interpolation settings of the respective instance.

localizedDesigner *string* Allows a language specific designer entry (name ID 9) with non-ASCII characters. Double click on the value, pick a language from the language menu, and enter the name. You can have multiple localizedDesigner parameters with different languages.

localizedFamilyName *string* Allows a language specific name with non-ASCII characters. Double click on the value, pick a language from the language menu, and enter the name. You can have multiple localizedFamilyName parameters with different languages. Will create encoded localized name table entries for IDs 1 and 4.

localizedStyleMapFamilyName *string* Language-specific variant of *styleMapFamilyName*. See that entry for further details.

localizedStyleName *string* Language-specific variant of the instance style name as entered in the user interface in *File > Font Info > Instances*. E.g., German 'Fett' or French 'Gras' for a Bold weight.

Make morx table *boolean* Inserts a 'morx' (extended glyph metamorphosis) AAT table into the exported font. Takes the contents of a prefix with the name 'morx' in *File > Font Info > Features*, written in MIF code (metamorphosis input file). For this to work, the *ftxenhancer* command line tool of the Apple Font Tools must be installed. For more information, refer to the documentation included with the Apple Font Tools.

makeOTF Argument *string* Semicolon-separated chain of *makeotf* Terminal arguments, including their respective dashes, e.g.,

You can download the
Apple Font Tools at
developer.apple.com/fonts

You can download the AFDKO at adobe.com/devnet/opentype/afdko

'-ns;-dcs;-ni'. For a documentation of available arguments, see the MakeOTF User Guide included in the AFDKO.

Master Background Color *color* Sets the canvas color of a master. The canvas assumes the specified color when the respective master is active in Edit view.

Master Color *color* For layered color fonts, sets the display color of the master (see 13.2, 'Layered Color Fonts', p. 152).

Master Name *string* Allows you to set a custom name for a master. This can be useful when the wording of the default options from the Weight and Width pop-ups may be misleading.

Name Table Entry *string* A custom entry for the OpenType name table. The syntax is one of the following three:

- <nameID>; <nameString>
- <nameID> <platformID>; <nameString>
- <nameID> <platformID> <encID> <langID>; <nameString>

If not specified, <platformID> will be assumed as 3, and successively, <encID> as 1 (Unicode), and <langID> as 0x0049 (Windows English). If only <platformID> is specified as 1, then both <encID> and <langID> will be assumed as 0 (Mac Roman, and Mac English).

The <nameID> can be anything except 1, 2, 3, 5, and 6, which cannot be set through this parameter. The <platformID> can either be 1 for Macintosh or 3 for Windows. The optional <encID> and <langID> represent either Windows or Macintosh encoding and language IDs, depending on the <platformID>. They must be numbers between 0 and 65536, and can be entered in decimal, octal or hexadecimal form. The AFDKO Syntax specification stipulates that 'decimal numbers must begin with a non-0 digit, octal numbers with a 0 digit, and hexadecimal numbers with a 0x prefix to numbers and hexadecimal letters a-f or A-F'

note *string* Arbitrary note about the font. This is not exported in the final OpenType font, only stored in the .glyphs file. Setting the font note as a custom parameter is equivalent to setting it in the Font Info UI.

openTypeHheaAscender *see hheaAscender*

openTypeHheaDescender *see hheaDescender*

openTypeHheaLineGap *see hheaLineGap*

openTypeNameCompatibleFullName *see compatibleFullName*

openTypeNameDescription *see description*

openTypeNameLicense *see license*

openTypeNameLicenseURL *see licenseURL*

For a list of possible values, see the specification for the OpenType name table: www.microsoft.com/typography/otspec/name.htm

openTypeNamePreferredFamilyName *see preferredFamilyName*

openTypeNamePreferredSubfamilyName *see*

preferredSubfamilyName

openTypeNameSampleText *see sampleText*

openTypeNameWWSFamilyName *see WWSFamilyName*

openTypeNameWWSSubfamilyName *see WWSSubfamilyName*

openTypeOS2Panose *see panose*

openTypeOS2Type *see fsType*

openTypeOS2TypoAscender *see typoAscender*

openTypeOS2TypoDescender *see typoDescender*

openTypeOS2TypoLineGap *see typoLineGap*

openTypeOS2UnicodeRanges *see unicodeRanges*

openTypeOS2VendorID *see vendorID*

openTypeOS2WeightClass *see weightClass*

openTypeOS2WidthClass *see widthClass*

openTypeOS2WinAscent *see winAscent*

openTypeOS2WinDescent *see winDescent*

openTypeVheaVertTypoAscender *see vheaVertTypoAscender*

openTypeVheaVertTypoDescender *see vheaVertTypoDescender*

openTypeVheaVertTypoLineGap *see vheaVertTypoLineGap*

Optical Size *string* Builds the Optical Size OpenType feature ‘size’, with encoded size menu names for Mac and Windows.

Requires a string with five semicolon-separated values:

- *design size*: size in decipoints (tenths of points) the font was designed for;
- *subfamily identifier*: arbitrary integer; different fonts with the same number can be grouped together in an optical size submenu, if the software supports it;
- *range start*: decipoint size of the size *above* which the font is supposed to be used for;
- *range end*: decipoint size of the size *until* (and including) which the font is supposed to be used for;
- *size menu name*: submenu name for the optical size, e.g., ‘Display’, ‘Subhead’, ‘Small’, or ‘Caption’.

Example: ‘100; 1; 69; 120; Ten’ will create a size feature that specifies 10 points as the intended design size, the range in which it is supposed to be used is 7 to 12 points, and the optical size name is ‘Ten’. Other fonts that use 1 as subfamily identifier and ‘Ten’ as name, can be grouped together.

panose *list* Once you click in the *Value* field, a dialog will appear that allows you to determine the setting for each category in the Panose specification. This corresponds to the OpenType

OS/2 table Panose field. “This 10 byte series of numbers is 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 values are fully described in the Panose “grey-book” reference, currently owned by Monotype Imaging. The PANOSE definition contains ten digits each of which currently describes up to sixteen variations. Windows uses bFamilyType, bSerifStyle and bProportion in the font mapper to determine family type. It also uses bProportion to determine if the font is monospaced. If the font is a symbol font, the first byte of the PANOSE number (bFamilyType) must be set to “pictorial.” At the time of this writing, PANOSE was hardly in use.

postscriptBlueScale *see blueScale*

postscriptBlueShift *see blueShift*

postscriptFontName *string* Name to be used for the FontName field in Type 1/CFF table. Should be ASCII-only, less than 30 characters long, and no whitespace allowed, e.g., ‘MyFont-BoldCdit’.

‘The FontName generally consists of a family name (specifically, the one used for FamilyName), followed by a hyphen and style attributes in the same order as in the FullName. For compatibility with the earliest versions of PostScript interpreters and with the file systems in some operating systems, Adobe limits the number of characters in the FontName to 29 characters. As with any PostScript language name, a valid FontName must not contain spaces, and may only use characters from the standard ASCII character set. If abbreviations are necessary to meet the 29 character limit, the abbreviations should be used for the entire family’ (Adobe Technote #5088).

Adobe recommends these abbreviations for style names: Bd Bold, Bk Book, Blk Black, Cm Compressed, Cn Condensed, Ct Compact, Dm Demi, DS Display, Ex Extended, Hv Heavy, Ic Inclined, It Italic, Ks Kursiv, Lt Light, Md Medium, Nd Nord, Nr Narrow, Obl Oblique, Po Poster, Rg Regular, Sl Slanted, Su Super, Th Thin, Up Upright. To further modify the above-mentioned names, you can prefix them with: Dm Demi, Sm Semi, Ult Ultra, X Extra.

postscriptFullName *string* Name to be used for the FullName field in Type 1/CFF table. This is the complete name of the font as it is supposed to appear to the user, and is thus allowed to

Adobe Technote #5088:
partners.adobe.com/public/
developer/en/font/
5088.FontNames.pdf

contain spaces, e.g., ‘My Font Bold Condensed Italic’.

Some systems match the family name ‘against the FullName for sorting into family groups.’ Therefore, the family name ‘must match the corresponding portion of the FullName, and be suitable for display in font menus. All fonts that are stylistic variations of a unified design should share the same FamilyName. [...] The FullName begins with a copy of the FamilyName and is completed by adding style attributes — generally in this sequence: weight, width, slope, optical size’ (Adobe Technote #5088).

postscriptIsFixedPitch *see isFixedPitch*

postscriptUnderlinePosition *see underlinePosition*

postscriptUnderlineThickness *see underlineThickness*

Post Table Type *integer* Version of the post table built into the instance, the default is 2 for TTF, and 3 for CFF fonts.

preferredFamilyName *string* Preferred family name. Corresponds to the OpenType name table name ID 16. ‘For historical reasons, font families have contained a maximum of four styles, but font designers may group more than four fonts to a single family. The Preferred Family allows font designers to include the preferred family grouping which contains more than four fonts. This ID is only present if it is different from ID 1’, the Family Name as set in Font Info.

preferredSubfamilyName *string* Preferred subfamily name. Corresponds to the OpenType name table name ID 17. ‘Preferred Subfamily; Allows font designers to include the preferred subfamily grouping that is more descriptive than ID 2. This ID is only present if it is different from ID 2 and must be unique for the Preferred Family.’

PreFilter *string* Same as *Filter*, but applied before decomposition. See its entry in the list of custom parameters in the Appendix (p. 184).

Preview Ascender *float* Master parameter for the distance between baseline and the upper edge of the preview in font units. Useful for scaling the preview at the bottom of the Edit View or in the Preview Panel when you have large ascenders that would otherwise be cut off. The default is 1000.

Preview Descender *float* Similar to *Preview Ascender*, a master parameter for the distance between baseline and the lower edge of the preview in font units. Defaults to *winAscent* if present, or otherwise, the *Descender* value set in *File > Font Info > Masters*.

- Reencode Glyphs** *list* Takes a comma-separated list of 'glyphname=unicodevalue' pairs, e.g., 'smiley=E100, logo=E101'. The parameter assigns the Unicode value to the glyph with the specified name at export. Should the Unicode value in question already be assigned to another glyph, the Unicode value of that other glyph will be deleted, but all production names will remain intact. It will remove a glyph's Unicode assignment if the Unicode value is left out, e.g., 'f_f_i=, f_f_j=' will strip f_f_i and f_f_j from their Unicode value.
- Remove Classes** *list* Prevents the export of the OpenType classes mentioned in the list. Useful for removing manually written classes when glyphs are removed from the font through the subsetting parameters. Note that automatic classes are removed automatically at export if the triggering glyphs are not in the font anymore, e.g., because they were removed or renamed with parameters.
- Remove Features** *list* Prevents the export of the OpenType features mentioned in the list. Useful when a glyph name suffix triggers Glyphs to generate a feature you do not want in the font, or you just want to disable a manually added feature for an instance. Note that automatic features are removed automatically at export if the triggering glyphs are not in the font anymore.
- Remove Glyphs** *list* Will prevent the glyphs mentioned in the list from being exported into the font. Automatically generated OpenType features respect changes in the glyph structure, e.g., if you remove all smallcap glyphs, then it will not auto-generate the smcp or c2sc features. Useful for subsetting.
- Remove post names for webfonts** *boolean* Removes glyph names in the webfont export, resulting in smaller file sizes.
- Rename Glyphs** *list* Will exchange the glyphs mentioned in the value with each other. Takes a list of rename strings of the form 'oldname=newname', e.g. 'e.bold=e, g.alt=g'. The glyph previously stored as newname will now be called oldname and vice versa. The parameter will update composites that employ the glyphs involved, update automatic features where necessary, and also exchange the *Exports* attributes of glyphs. If you want to avoid the export of one the glyphs, make sure that either their *Exports* attributes are set accordingly, or use the *Export Glyphs* parameter.
- Replace Class** *string* Replaces OpenType class code with custom code. The first word must be the class name (without the at

sign), followed by a semicolon, and the new class code. Works only if the class exists in *File > Font Info > Features*. This is only necessary for manually set up classes. Automatically generated classes update automatically.

Replace Feature *string* Replaces the content of an OpenType feature with the code specified. The first four letters must be the feature name (such as 'liga'), followed by a semicolon and the new feature code. Works only if the feature exists in *File > Font Info > Feature*.

ROS *string* Sets the ROS (Registry, Ordering, Supplement) for fonts with a Character To Glyph Index Mapping Table (cmap).

Available values are the public ROSES:

- Adobe-CNS1-6
- Adobe-GB1-5
- Adobe-Japan1-3
- Adobe-Japan1-6
- Adobe-Korea1-2
- Adobe-Identity-0

If you use 'Adobe-Identity-0', a GSUB table will be generated from the available OpenType features. Otherwise, the cmap and GSUB resources supplied by Adobe are used.

sampleText *string* Sample text. Corresponds to the OpenType name table name ID 19. 'This can be the font name, or any other text that the designer thinks is the best sample to display the font in.' This sample text is displayed, for instance, by Apple Font Book, when the font is selected in Sample view.

Save as TrueType *boolean* Exports the instance as TTF, regardless of the settings in the Export dialog.

Scale to UPM *integer* Scales the whole font to the supplied integer value. This is useful for scaling to a UPM of 2048 (or a power of two between 16 and 16,384) for TTF export, or if you are designing in an UPM size other than the default 1000.

shoulderHeight *integer* A vertical metric value for Arabic, Hebrew, and Indic scripts. Will be displayed in Edit view instead of the x-height in scripts that need a vertical metric line different from the x-height.

smallCapHeight *integer* A vertical metric for small caps. The algorithm for automatic creation of alignment zones respects this value. When a small cap glyph is displayed in Edit view and metrics are set to show, the small cap height will be displayed instead of the x-height.

styleMapFamilyName *string* Family name used for bold, italic, and bold italic style mapping. You can use this to create subfamilies within larger font families. 'Up to four fonts can share the Font Family name, forming a font style linking group.' Glyphs uses the entries in Style Name field and in the Style Linking section in the Instances tab of the Font Info for linking the four individual weights.

trademark *string* Trademark statement. Corresponds to the OpenType name table name ID 7. According to Microsoft, 'this is used to save any trademark notice / information for this font. Such information should be based on legal advice. This is distinctly separate from the copyright.'

TrueType Curve Error *float* Maximum deviance of the approximated TrueType curve in units. Default is 0.6. A higher curve error allows the TrueType converter to use fewer quadratic splines to approximate the cubic splines of your design. This can result in a significantly smaller 'glyph' table (containing the quadratic outline data), and smaller overall file size.

TTFAutohint binary path *string* File path to a precompiled TTFAutohint binary that should be used instead of the built-in TTFAutohint. This can be useful if you need to stick to a specific version or want to employ a newer version of TTFAutohint than Glyphs incorporates.

TTFAutohint control instructions *string* This allows you to specify TTFAutohint control instructions. It is recommended to prepare the control code in a separate file and then paste it into the value of the parameter. Possible instructions are:

- <glyph> left <pointIDs> <offset>
- <glyph> right <pointIDs> <offset>
- <glyph> nodir <pointIDs>
- <glyph> touch <pointIDs> xshift <x> yshift <y> @ <PPMs>
- <glyph> point <pointIDs> xshift <x> yshift <y> @ <PPMs>

Values for <offset> are optional and assumed as zero when omitted. In the touch and point instructions, either or both of the shifts can be specified. <x> and <y> must be between 0.0 and 1.0. <glyph> can be one or more comma-separated glyph names, specified as production names. <PPMs> can be a single PPM size, a size range of PPMs with a hyphen, or a comma-separated list of sizes and size ranges. A line that starts with a hashtag (#) is considered a comment and therefore ignored. The instructions can be abbreviated with their respective first letters, e.g., 'right' can be written as 'r'.

For more details about the control file syntax, see the TTFAutohint documentation on freetype.org/ttfautohint

TTFAutohint options *string* Specifies commandline options for the TrueType autohinter 'ttfautohint'. Use the dialog sheet to configure your settings:

For an in-depth discussion of TTFAutohint, see: freetype.org/ttfautohint/doc/ttfautohint.html
All direct quotes are from the ttfautohint Introduction.

- *Hint Set Range*: the PPM range for which the instructions will be optimized. Large ranges can cause huge file sizes.
- *Default Script*: 'default script for OpenType features'.
- *Fallback Script*: 'default script for glyphs that can't be mapped to a script automatically'.
- *Hinting Limit*: the PPM size 'where hinting gets switched off'. Default is 200 pixels, must be larger than the maximum of the hint set range. Pixel sizes up to this size use the hinting configuration for the range maximum.
- *x-Height Increase Limit*: from this pixel size down to 6 PPM, the x-height is more likely to be rounded up. Default is 14 PPM. 'Normally, ttfautohint rounds the x height to the pixel grid, with a slight preference for rounding up. (...) Use this flag to increase the legibility of small sizes if necessary.' Set to 0 if you want to switch off rounding up the x-height.
- *x-Height Snapping Exceptions*: 'list of comma-separated PPM values or value ranges at which no x-height snapping shall be applied', e.g., '8, 10-13, 16' disables x-height snapping for sizes 8, 10, 11, 12, 13, and 16. An empty string means no exceptions, and a mere dash ('-') disables snapping for all sizes.
- *Windows Compatibility*: 'This option makes ttfautohint add two artificial blue zones, positioned at the usWinAscent and usWinDescent values (from the font's OS/2 table). The idea is to help ttfautohint so that the hinted glyphs stay within this horizontal stripe since Windows clips everything falling outside.' Use this option if clipping occurs in Microsoft Windows.
- *Pre-Hinting*: 'whether native TrueType hinting of the input font shall be applied to all glyphs before passing them to the autohinter. (...) Use this only if the old hints move or scale subglyphs independently of the output resolution, for example some exotic CJK fonts.'
- *Hint Composites*: 'By default, the components of a composite glyph get hinted separately. If this flag is set, the composite glyph itself gets hinted (and the hints of the components are ignored). Using this flag increases the bytecode size a lot, however, it might yield better hinting results.'
- *Symbol Font*: 'Process a font that ttfautohint would refuse otherwise because it can't find a single standard character for any of the supported scripts. For all scripts that lack proper

standard characters, `ttfautohint` uses a default (hinting) value for the standard stem width instead of deriving it from a script's set of standard characters. Use this option (usually in combination with option `Fallback Script`) to hint symbol or dingbat fonts or math glyphs, for example, at the expense of possibly poor hinting results at small sizes.'

- *Dehint*: Disables all TT hinting, and therefore overrides all other options. Use only for testing.
- *Add Autohint Info*: appends 'ttfautohint version and command line information to the version string or strings (with name ID 5) in the font's name table.'
- *Strong Stems*: specifies for which rendering targets to use strong stem hinting, which 'snaps both stem widths and stem positions to integer pixel values as much as possible, yielding a crisper appearance at the cost of much more distortion.'

Possible rendering targets are: *Grayscale* (Android), *GDI ClearType* (rasterizers v.36 and v.37, e.g. Win XP), *DW ClearType* (rasterizers v.38+, i.e., IE9+ and Win 7 and newer).

TTFStems *list* A list of horizontal stem definitions for TrueType only. For each horizontal stem, you can define a name and an average value. When you edit the value, a dialog sheet will drop down. Use the gear menu to add or remove stem definitions, or import the currently available horizontal PostScript stems from the *Horizontal Stems* field.

typoAscender *integer* The height of the ascenders in units. Corresponds to the OpenType OS/2 table `sTypoAscender` field. 'The typographic ascender for this font. Remember that this is not the same as the Ascender value in the `hhea` table, which Apple defines in a far different manner. [...] The suggested usage for `sTypoAscender` is that it be used in conjunction with `unitsPerEm` 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. These metrics will be exposed through Windows APIs. Macintosh applications will need to access the `sfnt` resource and parse it to extract this data from the "OS/2" table.

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 `sTypoAscender` is that which describes the top of the of the ideographic em-box. For example, if the ideographic em-box of the font extends from coordinates 0, -120 to 1000,880 (that is, a 1000 × 1000 box set 120 design units below the Latin baseline), then the value of `sTypoAscender` must be set to 880. Failing to adhere to these requirements will result in incorrect vertical layout.’

typoDescender *integer* The depth of the descenders in units (negative value). Corresponds to the `sTypoDescender` field of the OpenType OS/2 table.

‘The typographic descender for this font. Remember that this is not the same as the Descender value in the `hhea` table, which Apple defines in a far different manner. One good source for `sTypoDescender` in Latin based fonts is the Descender value from an AFM file. For CJK fonts see below.

The suggested usage for `sTypoDescender` is that it be used in conjunction with `unitsPerEm` 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. These metrics will be exposed through Windows APIs. Macintosh applications will need to access the `sfnt` resource and parse it to extract this data from the “OS/2” table (unless Apple exposes the “OS/2” table through a new API).

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 `sTypoDescender` is that which describes the bottom of the of the ideographic em-box. For example, if the ideographic em-box of the font extends from coordinates 0,-120 to 1000,880 (that is, a 1000 × 1000 box set 120 design units below the Latin baseline), then the value of `sTypoDescender` must be set to -120. Failing to adhere to these requirements will result in incorrect vertical layout.’

typoLineGap *integer* The recommended whitespace between lines, measured in units. Corresponds to the OpenType OS/2 table `sTypoLineGap` field. ‘The typographic line gap for this font. Remember that this is not the same as the LineGap value in the `hhea` table, which Apple defines in a far different manner.

The suggested usage for `sTypoLineGap` is that it be used in

conjunction with `unitsPerEm` to compute a typographically correct default line spacing. Typical values average 7–10% of units per em. 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.'

`underlinePosition` *integer* The suggested distance from the baseline to the top of the underline. Negative values indicate a position below the baseline. Corresponds to the CFF table entry `UnderlinePosition`. Default is `-100`.

`underlineThickness` *non-negative integer* Underline thickness value. Corresponds to the CFF table entry `UnderlineThickness`. Default is `50`.

`unicodeRanges` *list* A list of supported Unicode ranges in the font. Corresponds to the OpenType OS/2 table `ulUnicodeRange1`, `ulUnicodeRange2`, `ulUnicodeRange3` and `ulUnicodeRange4` fields. Use this parameter to override the automatic setting by the app. The dialog offers a search field, so you can quickly spot the proper ranges for your fonts. E.g., if you want to cover all Latin ranges, simply search for 'latin' and all corresponding ranges in the list will be displayed.

`uniqueID` *non-negative integer* PostScript language level 1 `UniqueID`. An optional, uniquely identifying integer between 0 and 16777215, stored in the PostScript Private Dictionary. From the PostScript Type1 specification: 'Its primary purpose is uniquely identifying bitmaps already created and cached from that font program; having a `UniqueID` allows the PostScript interpreter to cache bitmaps across jobs.' The entry is deprecated, but remains in use for CJK fonts. Adobe discourages its use, especially in non-CJK fonts. Use this only if you know what you are doing.

`unitsPerEm` *non-negative integer* Units per em. Default is 1000 for PostScript-flavored OpenType fonts and a power of two between 16 and 16,384 (usually 2048) for TrueType-flavored OpenType fonts. The value specified is the amount of units that will be used for the font size. A smaller value will cause the font to appear larger on screen, and vice versa. This parameter will only set the UPM value, and not scale node coordinates and other measurements. If you do want to scale, see *Scale to UPM*.

Update Features *boolean* Forces an update of all automatic feature code. This is especially useful in a phase of font production where the glyph set changes a lot, or for suppressing the automatic feature code generation.

Use Extension Kerning *boolean* If checked, additional kern lookups will be created with a GSUB or GPOS Extension lookup type, allowing the font to store more kerning values. Use this when the attempt to export your font results in an offset overflow error, and you cannot or do not want to delete kern pairs, especially exceptions. While all modern Office and Layout software supports extended kerning, this option may render the kerning data in your font incompatible with old software.

Use Line Breaks *boolean* If checked, line breaks inside OpenType features will not be escaped (i.e., replaced with `\012`) when stored in a `.glyphs` file. This can facilitate version control.

Use Typo Metrics *boolean* If checked, applications that respect this setting (in particular, versions of Microsoft Office since 2006) will prefer `typoAscender`, `typoDescender`, and `typoLineGap` over `winAscent` and `winDescent` for determining the vertical positioning. Default is off. Corresponds to bit 7 ('don't use Win line metrics') in the OS/2 table `fsSelection` field. According to the MakeOTF User Guide, this bit was introduced 'so that reflow of documents will happen less often than if Microsoft just changed the behaviour for all fonts.'

vendorID *string* Four character identifier for the creator of the font. Corresponds to the OpenType OS/2 table `achVendID` field. If not set, Glyphs will use 'UKWN' ('unknown') as Vendor ID. 'The four character identifier for the vendor of the given type face. This is not the royalty owner of the original artwork. This is the company responsible for the marketing and distribution of the typeface that is being classified. It is reasonable to assume that there will be 6 vendors of ITC Zapf Dingbats for use on desktop platforms in the near future (if not already). It is also likely that the vendors will have other inherent benefits in their fonts (more kern pairs, unregularized data, hand hinted, etc.). This identifier will allow for the correct vendor's type to be used over another, possibly inferior, font file. The Vendor ID value is not required.'

versionString *string* A placeholder string into which the version number will be inserted automatically, e.g., 'Version %d.%03d', where %d stands for an integer, and %03d for integer

Microsoft keeps a list of registered vendors at: microsoft.com/typography/links/vendorlist.aspx

represented with three digits, e.g., '008'. Will be used to overwrite Name ID 5, the version string in the name table.

vheaVertTypoAscender *integer* Ascender value for vertical type-setting. Corresponds to the `vertTypoAscender` field in the OpenType `vhea` table.

vheaVertTypoDescender *integer* Descender value for vertical type-setting. Corresponds to the `vertTypoDescender` field in the OpenType `vhea` table.

vheaVertTypoLineGap *integer* Line gap value for vertical type-setting. Corresponds to the `vertTypoLineGap` field in the OpenType `vhea` table.

Webfont Formats *list* For the instance, in which this parameter is specified, the listed webfont formats will be exported, regardless of the settings in the Export dialog. Possible values: EOT, WOFF or WOFF2.

Webfont Only *boolean* If activated, it removes some of the information stored in the font file necessary for desktop use. This makes it harder to convert the webfont into a different format or to install it locally in an operating system like Windows or MacOS. Careful: Technically, this option produces a damaged font, which, however, still works as webfont in browsers.

weightClass *integer* Weight class value. Must be a non-negative integer. Corresponds to the `usWeightClass` field of the OpenType OS/2 table. 'Indicates the visual weight (degree of blackness or thickness of strokes) of the characters in the font.' Overrides the value set by the *Weight* drop-down list of the instance.

<i>Value</i>	<i>Description</i>
100	Thin
200	Extra-light (Ultra-light)
300	Light
400	Normal (Regular)
500	Medium
600	Semi-bold (Demi-bold)
700	Bold
800	Extra-bold (Ultra-bold)
900	Black (Heavy)

Some applications, such as Adobe apps, use this value to sort the subfamilies in the font menu. Interpretation of this value varies greatly though. There is a bug in some older Microsoft renderers which causes all fonts with a `weightclass` lower than 250 to be artificially boldened on screen. Other type

engines only consider the hundreds digit, and ignore the tens and units, i.e., they make no difference between 400 and 425. Some systems ignore the value completely.

widthClass *integer* Width class value. Must be in the range 1–9. Corresponds to the `usWidthClass` field in the OpenType OS/2 table. Some applications use this value to sort the subfamilies in the font menu. Overrides the value set by the *Width* dropdown list of the instance.

<i>Value</i>	<i>Description</i>	<i>% of normal</i>
1	Ultra-condensed	50
2	Extra-condensed	62.5
3	Condensed	75
4	Semi-condensed	87.5
5	Medium (normal)	100
6	Semi-expanded	112.5
7	Expanded	125
8	Extra-expanded	150
9	Ultra-expanded	200

‘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. 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.’

winAscent *non-negative integer* The top extremum of the font rendering box for Windows. Thus, `winAscent` should be high enough to include caps and their accents. Corresponds to the `usWinAscent` field in the OpenType OS/2 table.

‘`usWinAscent` is computed as the `yMax` for all characters in the Windows ANSI character set. `usWinAscent` is used to compute the Windows font height and default line spacing. For platform 3 encoding 0 fonts, it is the same as `yMax`. Windows will clip the bitmap of any portion of a glyph that appears above this value.’

winDescent *integer* The bottom extremum of the font rendering box for Windows (positive value). Thus, `winDescent` should be large enough to encompass the descenders of lowercase letters like *g*, *p*, *q*, and *y*. Corresponds to the `usWinDescent`

The term ANSI refers to the Windows 8-bit encoding 1252, which is described at msdn.microsoft.com/en-us/goglobal/cc305145 and covers mostly Western European Latin characters.

field of the OpenType OS/2 table. ‘usWinDescent is computed as the –yMin for all characters in the Windows ANSI character set. usWinDescent is used to compute the Windows font height and default line spacing. For platform 3 encoding 0 fonts, it is the same as –yMin. Windows will clip the bitmap of any portion of a glyph that appears below this value.’

Write Kern Table *boolean* On export, will write an old-style kern table in addition to the kern feature in the GPOS table. All group kerning will be expanded into all possible singleton pairs. This means that you will have to subset before using this parameter, otherwise you risk a table overflow.

WWSFamilyName *string* WWS family name. WWS stands for ‘Weight Width Slope’. Corresponds to the OpenType name table name ID 21. ‘Used to provide a WWS-conformant family name in case the entries for IDs 16 and 17 do not conform to the WWS model. (That is, in case the entry for ID 17 includes qualifiers for some attribute other than weight, width or slope.) [...] Examples of name ID 21: “Minion Pro Caption” and “Minion Pro Display”. (Name ID 16 would be “Minion Pro” for these examples.)’

WWSSubfamilyName *string* WWS Subfamily name. Corresponds to the OpenType name table name ID 22. ‘Used in conjunction with ID 21, this ID provides a WWS-conformant subfamily name (reflecting only weight, width and slope attributes) in case the entries for IDs 16 and 17 do not conform to the WWS model. [...] Examples of name ID 22: “Semibold Italic”, “Bold Condensed”. (Name ID 17 could be “Semibold Italic Caption”, or “Bold Condensed Display”, for example.)’ For name IDs 16 and 17, see the entries for preferredFamilyName and preferredSubfamilyName, respectively.

17.4 CHANGING THE GLYPH DATA

17.4.1 Global Glyph Data Changes

If you want to control which marks are associated with which base glyph or how a compound glyph is composed by default, you can write your own glyph data. To do this, create a copy of GlyphData.xml in the Info folder you can find via *Script > Open Scripts Folder*. Create it next to the Scripts folder if it is not there yet. You will find the default XML file in the Package Contents of the Glyphs application. To access it, right-click on the app icon in Finder, and pick *Show Package Contents*.

For a step-by-step walkthrough, see the tutorial ‘Roll Your Own Glyph Data’ on glyphsapp.com/tutorials/roll-your-own-glyph-data

Drill down to Contents/Frameworks/GlyphsCore.framework/Versions/A/Resources/GlyphData.xml, and copy it into the Info folder. The new XML file only needs the lines you want to customize, so you can delete all lines you want to stay the same. For changes in the Glyph Data XML to take effect, you need to restart the application.

The XML must have one *glyphData* element containing any number of *glyph* elements. Every *glyph* element must have these three compulsory attributes:

- *name*: the glyph name;
- *category*: the category of the glyph, as displayed in Font view or in *Window > Glyph Info*, e.g., letter or digit.

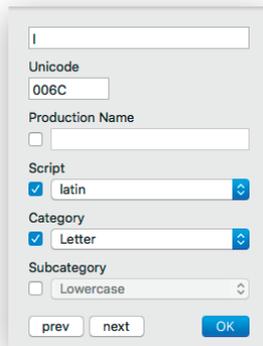
And it can have any of these optional attributes:

- *accents*: comma-separated list of glyph names describing the marks that appear in the mark cloud for the glyph in question;
- *altNames*: comma-separated list of alternate or legacy names that Glyphs recognizes for converting into the glyph name (as set in the required name attribute) via *Glyph > Update Glyph Info*;
- *anchors*: comma-separated list of anchor names describing anchors which are inserted by default when you choose *Glyph > Set Anchors* (Cmd-U);
- *decompose*: in compound glyphs, the glyph names of the components in the correct order, e.g., letter – mark, in a comma-separated list;
- *description*: a written description of the glyph, usually taken from the character information in the Unicode standard;
- *production*: the production name of the glyph, i.e., the name as it will be written into the OpenType font file;
- *script*: the script system to which the glyph belongs, e.g., arabic, latin, greek;
- *sortName*: the name by which a glyph is sorted in Font view, e.g., d1 if you want it to appear right after d;
- *sortNameKeep*: similar to *sortName*, but used when *File > Font Info > Other Settings > Keep Alternates Next to Base Glyph* is active;
- *subCategory*: a further specification of the category, e.g., ‘Uppercase’ or ‘Lowercase’ for letters;
- *unicode*: the Unicode value associated with the glyph, written as four- or five-digit hex string.

17.4.2 Local Glyph Data Changes

If you want to change glyph data for a single file only, you have multiple options. Firstly, you can keep a custom `GlyphData.xml` in the same folder as your `.glyphs` file, in an `Info` subfolder next to your `.glyphs` file, or in the parent folder above the `.glyphs` file. This way, multiple `.glyphs` files can share the same `GlyphData.xml`. When you pass on the `.glyphs` file with a third party, make sure the XML file travels along.

Secondly, you can change the glyph data only for specific glyphs inside a single `.glyphs` file. To do that, select one or more glyphs in Font view, and choose *Edit > Info for Selection* (Cmd-Opt-I). In the dialog that appears, activate the check marks for the fields you want to override, and edit the data accordingly. For a single glyph, you can edit glyph name, Unicode value, production name, script, category, and subcategory. For a selection of multiple glyphs, you can simultaneously set script, category, and subcategory.



17.5 CHANGING TOOL SHORTCUTS

You can change a shortcut for a tool in the toolbar via a Terminal command. Start Terminal.app (located in `/Applications/Utilities`), and type:

```
defaults write com.GeorgSeifert.Glyphs2 <tool> "<key>"
```

To reset the shortcut back to its default, type:

```
defaults delete com.GeorgSeifert.Glyphs2 <tool>
```

In either case, confirm by pressing the Return key. The change should be effective immediately, i.e., you do not need to restart the application. For `<key>`, pick any lowercase letter you can type with a single push of a key on the keyboard. For `<tool>`, use any of these:

- AnnotationTool.Hotkey (default: a)
- DrawTool.Hotkey (p)
- HandTool.Hotkey (h)
- MeasurementTool.Hotkey (l)
- OtherPathsTool.Hotkey (e)
- PenTool.Hotkey (b)
- PrimitivesTool.Hotkey (f)
- SelectTool.Hotkey (v)
- SelectAllLayersTool.Hotkey (v)
- TextTool.Hotkey (t)
- RotateTool.Hotkey (r)
- ScaleTool.Hotkey (s)
- TrueTypeTool.Hotkey (i)
- ZoomTool.Hotkey (z)

The hotkey IDs correspond to the tool names in the UI, except for OtherPathsTool.Hotkey, which controls the shortcut for the Knife and Eraser tools. E.g., to set a new shortcut for the Annotation tool (default shortcut A), you would start Terminal.app, and type, on one line:

```
defaults write com.GeorgSeifert.Glyphsz
AnnotationTool.Hotkey "q"
```

Type the letter corresponding to the key in lowercase. Confirm with the Return key. Now the Annotation tool can be invoked by pressing the Q key. If you want to reset it back to its default, you would type, again on a single line:

```
defaults delete com.GeorgSeifert.Glyphsz
AnnotationTool.Hotkey
```

And again, press Return to confirm the entry. Now the tool shortcut is the A key again.