

Keyboard Maestro 6 Documentation

[Overview](#)

[What's New](#)

[Features](#)

[Purchase](#)

[Screenshots](#)

[Tour](#)

[Links](#)

[Quick Start](#)

[How do I ...?](#)

- [How do I install Keyboard Maestro?](#)
- [How do I upgrade Keyboard Maestro?](#)
- [How do I purchase Keyboard Maestro?](#)
- [Can I purchase Keyboard Maestro from the Mac App Store?](#)
- [How do I register Keyboard Maestro?](#)
- [How do I get started?](#)
- [How do I create a new Macro?](#)
- [How do I find a Macro I've used or modified recently?](#)
- [How do I cancel a running Macro?](#)
- [How can I get the mouse coordinates on the screen or in a window?](#)
- [How do I insert styled/colored text or images?](#)
- [How do I Insert the current date?](#)
- [How do I get more than one Macro Palette?](#)
- [How do I use a multiple keystroke trigger?](#)
- [How do I configure the Application Switcher?](#)
- [How do I uninstall Keyboard Maestro?](#)
- [How do I revert to a previous version of Keyboard Maestro?](#)

[Macro Groups](#)

[Macros](#)

[Macro Triggers](#)

- [Overview](#)
- [Hot Key](#)
- [Typed String](#)
- [Application](#)
- [Login](#)
- [Engine Launch](#)
- [System Wake](#)
- [Time](#)
- [While Logged In](#)
- [Macro Palette](#)
- [Status Menu](#)
- [Public Web](#)
- [Mounted Volume](#)
- [USB Device](#)
- [Wireless Network](#)
- [Device Key](#)
- [MIDI Note](#)
- [By Script](#)

[Macro Actions](#)

- [Overview](#)
- [Application Control](#)
- [Clipboard Actions](#)
- [Control Flow Actions](#)
- [Debugger Actions](#)
- [Mail Control Actions](#)
- [Execute Actions](#)
- [File Actions](#)
- [Safari and Google Chrome Actions](#)
- [Image Actions](#)
- [Interface Control](#)
- [iTunes Control](#)
- [Keyboard Maestro Actions](#)
- [MIDI Actions](#)
- [Notification Actions](#)
- [Open Actions](#)
- [QuickTime Player Control](#)
- [Switcher Actions](#)
- [System Control](#)
- [Text Actions](#)
- [Variable Actions](#)
- [Prompt For User Input Action](#)
- [Web Actions](#)

[Macro Syncing](#)

[Macro Debugger](#)

[Variables](#)

[Filters](#)

[Text Tokens](#)

[Calculations](#)

[Conditions](#)

[Recording](#)

[Macro Library](#)

[Macro Examples](#)

- [Launch Your Most Used Applications](#)
- [Open Your Most Used Documents](#)
- [Insert Text Templates](#)
- [Use Hot Keys to Open Financial Accounts](#)
- [Use Hot Keys to Connect to SSH or FTP sites](#)
- [Simulate Bookmarks](#)
- [Remap Command Keys](#)
- [Simulate Missing Features](#)
- [Swap Characters](#)
- [Save a Text Clipping](#)
- [Delayed Click](#)
- [Insert Boilerplate Text](#)
- [Apply Text Conversions](#)
- [Simulate Workspaces](#)
- [Setup an Application When Launched](#)
- [Clean Up After Using an Application](#)
- [Launch Scanner Application When Scanner is Connected](#)
- [Switch Network Location When You Connect](#)
- [Feedback During Macro Execution](#)
- [Rakesh Kumar's PC Switcher's Pack](#)

[Icon Chooser](#)

[Application Launcher](#)

[Application Switcher](#)

[Window Switcher](#)

[Clipboard Switcher](#)

[Clipboard History Switcher](#)

[Preferences](#)

- [General Preferences](#)
- [Web Server Preferences](#)
- [Clipboards Preferences](#)
- [Variables Preferences](#)
- [Excluded Preferences](#)
- [Other Hidden Preferences](#)

[Scripting](#)

- [Executing Scripts](#)
- [Controlling Keyboard Maestro via Scripting](#)
- [Enhancing AppleScript](#)
- [The keyboardmaestro URL Scheme](#)

[Status Menu Icons](#)

[Plug In Actions](#)

[Windows](#)

- [Welcome Window](#)
- [Macros Window](#)
- [Tutorial](#)
- [Macro Group Editor](#)
- [Macro Editor Window](#)
- [Macro Library Window](#)
- [Icon Chooser Window](#)
- [Recording Window](#)
- [Macro Debugger](#)
- [Conflict Palette](#)
- [Trigger Macro by Name](#)
- [Application Launcher Window](#)
- [Application Switcher Window](#)
- [Window Switcher Window](#)
- [Clipboard Switcher Window](#)
- [Clipboard History Switcher Window](#)
- [Preferences Window](#)
- [Preferences General Pane](#)
- [Preferences Web Server Pane](#)
- [Preferences Variables Pane](#)
- [Preferences Clipboards Pane](#)
- [Preferences Exclude Pane](#)
- [About Window Pane](#)

[Menus](#)

- [Keyboard Maestro](#)
- [File](#)

- [Edit](#)
- [View](#)
- [Window](#)
- [Help](#)
- [Status Menu](#)

[Tips](#)

[Troubleshooting](#)

[Support](#)

[Glossary](#)

[Administrative Details](#)

- [Requirements](#)
- [Distribution](#)
- [History](#)
- [Credits](#)
- [Warranty](#)
- [Licenses](#)
- [Fine Print](#)

Overview

Keyboard Maestro will take your Macintosh experience to a new level. Keyboard Maestro enables you to create or record custom macro shortcuts that you can activate at any time. For example, your macros could help you navigate running applications or work with an unlimited number of clipboards. Best of all, every macro you create is available using simple keystrokes you choose. The only limit to Keyboard Maestro is your imagination!

Using Keyboard Maestro's powerful [Macros](#), you can make your Mac behave the way you want it to behave – open documents when and where you want them, type sentences with the press of a key, expand abbreviations into entire paragraphs, control web applications, and much more. You simply define what you want your Mac to do and when you want it done.

Keyboard Maestro comes complete with a [clipboard history](#), saving everything you copy for later use so you'll never lose something on your clipboard again, as well as [named clipboards](#) where you can store commonly used images or text.

Keyboard Maestro also includes a powerful [Application Switcher](#) and [Window Switcher](#) so you can cycle through applications or windows, closing, hiding, launching, and more as well as an [Application Launcher](#) that lets you quickly launch applications.

Keyboard Maestro requires an Intel Mac running [Mac OS X 10.8](#) or later.

Keyboard Maestro is free to try with no limitations. Once the trial period ends, a [license must be purchased](#) to continue using it.

What's New

Keyboard Maestro 6 expands on the powerful base of previous versions.

Changes for 6.4.8

- Workaround Yosemite bug in Speak Text action.
- Workaround Yosemite bug in display of popup menus.
- Fixed Yosemite display of Welcome screen.
- Fixed Yosemite freeze with network reachability (typically after sleep).
- Add Help button with information about Yosemite Accessibility Permission bugs.
- Limit Undo stack size.
- Add support for Corsair K95.
- Allow up to 9999 Seconds/Minutes/Hours in While Logged In Trigger.
- Fixed behaviour of Image Actions on Retina Macs.
- Fixed misspelled Accessibly.
- Fixed misspelled "Pixek Height" in Get Image Size action.

Changes for 6.4.7

- Fixed a drawing issue in the Clipboard History in Yosemite DP6.
- Fixed a crash in the editor.

Changes for 6.4.6

- Fixed the Fast User Switch action.

Changes for 6.4.5

- Fixed a crash in Trigger Macro by Name

Changes for 6.4.4

- Added Command-Option-R Record Without Delay menu.
- Improved reliability of Typed String triggers with Conflict Palette.
- Condition updates now work properly when there were no conditions.
- Worked around a bug in Pages that made styled text paste as plain text.
- Worked around a system issue with launching shell scripts and some environment variables.
- Resolved a drawing glitch in Yosemite.
- Removed the use of some newly deprecated APIs.
- Fixed excess memory/CPU usage when a screen condition and a second condition were used together.
- Fixed a possible crash in the calculation engine.
- Fixed a possible crash saving the Time Saved.
- Fixed the inverted checkboxes in Quit/Hide All/Other Applications.

Changes for 6.4.3

- Resolved a problem with the editor using too much CPU, especially with Chrome actions.
- The editor now only updates visible actions (actions actually visible in the window).
- Fixed incorrect text in non-edit mode display of Quit/Hide Applications.

Changes for 6.4.2

- Added accessibility labels/tool tips for Add/Delete Variable/Clipboard buttons in Preferences.
- Composite Image text now processes text tokens.
- Allow the Escape key to close the preferences window.
- Worked around an issue with Get Info in the Finder.
- Stop tag actions from crashing in Mountain Lion.
- Fixed the displayed length of checkboxes in Prompt For User Input.
- Quit or Hide Applications action inverts the Exclude checkbox when setting.
- Avoided another crash involving CFPReferences.
- Disable Secure Input Mode warning in the first two minutes after startup.
- Fixed a crash in `unlockViewHierarchyForDrawing`.
- Fixed potential crashes caused by `NSAppleScript`.
- Fixed potential crash if a volume is unmounted as Keyboard Maestro Engine quits.
- Fixed a potential crash in the editor.
- Removed bogus "Reload" contextual menu from WebKit views.
- Removed overly enthusiastic `isHiddenOrHasHiddenAncestorOrIsNotInWindow` assertions.
- Undo now correctly reselects selected actions.

Changes for 6.4.1

- Fixed typo in Get File Attribute editor.
- Fixed an issue where simulated text could trigger Typed String triggers.
- Fixed a case where Application Quit trigger for background apps might not fire.
- Treat smart and straight quotes as equivalent for menu/button matching.
- Non-edit mode display now updates for programatically changed enables.
- Fixed an issue with some hot keys in palettes opened by duplicate palettes.
- Fixed a potential crash when creating custom icons without the Apple Color Emoji font.
- Added For Each examples for Finder selection and folder contents to action selector.
- Stopped Web Search from performing search if it lost focus.
- Fixed a possible crash processing text tokens.
- Added accessibility labels for various unlabeled image buttons in palettes.
- Fixed an issue where timer drift could cause time triggers to execute twice.
- Fixed a crash undoing Execute AppleScript actions.
- Changed "iPhone" to "iOS" in preference references.
- Fixed an issue in Exit From Loop that would partially cancel following actions.

Changes for 6.4

- Added support for Mavericks Tags.
- Added Get File Attribute: Tags. Mavericks only.
- Added Set File Attribute: Tags (set, add, toggle, or remove). Mavericks only.
- Added Semaphore Lock/Unlock actions.
- Added Asynchronous option to Execute Macro

- Added Asynchronous option to Execute Shell Script, AppleScript and JavaScript.
- Improved VoiceOver support for lists and buttons.
- Improved accessibility support.
- Added speech rate to the Speak Text action.
- Added importMacros AppleScript command.
- Added deleteMacro and deleteMacroGroup AppleScript commands.
- Added Tab to the hot key/simulate keystroke popup menus.
- Added %ExecutingMacroUID% token.
- Added Mouse, Front, Back, Back2 options to the SCREEN function.
- Added TRIGGERTIME() function.
- Added GMTOFFSET() function.
- Added %AddressBook%Note% text token.
- Added SimulateKeystrokeDeadKeyDelay private preference.
- Added various Cancel action varieties to the action selector.
- Adjusted Typed String trigger to make diacritical sensitivity independent of case sensitivity.
- SCREENSAVER() returns true if screen is locked.
- Set %ActionResult% on timeout failures.
- Allow selection of .xpc applications.
- Changed Keychain access to use direct code instead of security utility.
- Fixed a crash when evaluating expressions of the form "("
- Stop clicks while deferred selection change pending, perhaps fix editor crashes.
- Fixed "Any Application" launch/quit trigger to only include foreground apps.
- Added macro/action to Macro Cancelled message.
- Add Help menu items for Regular Expression and ICU Date Time reference.
- Fixed a potential crash in the window switcher setup.
- Fixed a potential crash when launching shell scripts.
- Fixed a potential crash finding buttons with bogus accessibility information.
- Fixed a memory leak in the image finding code.
- Fixed a potential crash showing the contact sheet.
- Fixed issue where time triggers would not fire if any changes were made within 30 seconds before the scheduled time.
- Fixed an issue with continuous timer triggering during daylight savings change over.
- Fixed a potential crash with action editors.
- Adjusted a few names in the action selector.
- Fixed Accessibility Alert display in Mountain Lion.
- Worked around a potential crash in the system preferences settings.
- Fixed an issue with non-edit display of Repeat actions.
- Fixed an issue with non-integral sized images (seriously!).
- Fixed an issue with shell scripts that waited for stdin input.
- Offer Getting Started emails even to registered users.
- Stop Time Saved for automatic (from editor) actions.
- Conflict Palette correctly highlights selection characters for macro names with sorting codes.
- Forcefully disabled dash and quote substitution in Insert Text and Scripts action editors.
- Word count treats contracted (eg I've) words as a single word.
- Resolved a problem typing some characters on some keyboards (eg ` on Swiss German keyboards).
- Resolved a problem with double clicking in the Finder.
- Resolve a problem with repeating triggers and Show Status Menu.
- Pause action merges together when recording pauses.

Changes for 6.3.2

- Fixed a code signing issue.

Changes for 6.3.1

- Allow more than one separator in Prompt For User Input menus.
- Fixed clipboard garbage text that affected some applications.
- Fixed Set Find Pasteboard action.
- Fixed a potential crash when executing scripts.
- Fixed a crash in the editor if a macro contained only an Execute This Macro action.
- Fixed a potential crash writing clipboard strings.
- Fixed a potential crash if getting the icon of a file failed.

Changes for 6.3

- Added Typed String trigger "Diacriticals do not matter" option.
- Add Typed String Trigger "regular expression match".
- Add Typed String Trigger "only after word break" option.
- Typed String simulates deletes before displaying the conflict palette.
- Added TypedStringClearTime private preference (default 5 seconds).
- Added TypedStringClearWithShiftSpace private preference (default on).
- Typed String triggers do not clear the Typed String buffers (simulated characters do).
- Added Clear Typed String Buffer action
- Added Crash Reporter.
- Added Substrings In collection.

- Merged Lines In collections.
- Added Lines In Named Clipboard collection.
- Added %ActionResult% token.
- Removed Press Button Result, Mouse Click Result, Select Menu Result variables.
- Added %Trigger%, %TriggerBase%, %TriggerValue% tokens.
- Removed USB Device Name, Mounted Volume Name variables.
- Added delay before and after dragging in the drag mouse action.
- Vastly improved performance of actions that write named clipboard entries.
- Changed various informative alert sheets to use notifications instead.
- Highlight errors in calculations in more places.
- Added option to Fix Finder Selection Bug in appropriate collections and actions.
- Added support for separators (-) in Prompt For User Input popup menus.
- Added support for code_display formatted menu items in Prompt For User Input popup menus.
- Use minus sign (-) instead of no entry sign (\) for Delete.
- Periodically trim log files.
- Added URL for keyboardmaestro://m=Activate%20Application%20Switcher
- Added URL for keyboardmaestro://q=Activate
- Added URL for keyboardmaestro://g=All%20Macros/q=Activate
- Added URL for keyboardmaestro://a=Execute
- Added URL for keyboardmaestro://c=All%20Actions/a=Execute
- Hold Command key while pasting to toggle Close On Action in Clipboard Switchers.
- Added support for displaying time saved in Days, Months or Years.
- Added SYSTEMVOLUME() function.
- Only force setting the Find Pasteboard if it actually changes value.
- Added support for Option-Arrow keys in switcher grid view.
- Allow more use of shortcut letters in switchers with non-English keyboards.
- Added warning to Status Menu if Accessibility is disabled for Engine.
- Reduced apparent Energy impact by reducing polling for clipboard changed.
- Added getall option to gethotkeys AppleScript engine command.

Bug fixes for 6.3

- Use a more modern method to simulate hardware keys which should resolve the Volume Control action issues.
- Wireless triggers would fire even if the macro was disabled or inactive.
- Save empty calculations even though they are invalid.
- Fixed Substring action when source and dest are different named clipboards.
- Fixed an issue with MIDI notes and non-zero channels.
- Removed Text Suite from AppleScript dictionary.
- Mouse Drag locked the engine up while dragging.
- Fixed potential crash when when Open/Close CDROM action fails.
- Screen image highlighting worked only on one screen in Mavericks.
- Fixed SCREEN(Internal,coord) and SCREEN(External,coord).

Changes for 6.2

- Added action to Send Mail Messages.
- Added action to Set Mail Flag/Flagged/Read/Junk status.
- Added Mail recipient tokens: MailRecipients, MailToRecipients, MailCCRecipients, MailBCCRecipients
- Added Mail tokens: MailSender, MailReplyTo, MailSubject, MailContents, MailRawSource
- Added Mail date functions: MAILDATERECEIVED(), MAILDATESENT()
- Added Mail status functions: MAILFLAG(), MAILFLAGGED(), MAILREADSTATUS(), MAILJUNKSTATUS()
- Added Mail action functions: MAILWASFORWARDED(), MAILWASREDIRECTED(), MAILWASREPLIEDTO()
- Added Reveal a File action.
- Format AppleScript in the Execute AppleScript action.
- Added menu condition to test for menus with a specified shortcut.
- Execute JavaScript scripts can now access Keyboard Maestro variables.
- Added Destination selection to Substring action.
- Extend Get Image Size to return size in pixels, points or DPI.
- Added Absolute Position relative for Mouse Click action.
- Image View supports Quick Look in Find Image, Mouse Click and Image on Screen condition.
- Add detail to macro edit message "Triggered by any of the following"
- Added %FindPasteboard% text token.
- Hold the option key down in the Application Switcher to reopen windows.
- Updated Markdown library from Andreas Zeitler (Mac OS X Screencasts) (requires Wrap Text plugin action).

Bug fixes for 6.2

- Resolved an issue where Keyboard Maestro Engine could not display any windows after login.
- Remove "Hide" flag from Keyboard Maestro Engine login item if present.
- Removed and correct any extraneous Keyboard Maestro Engine login items.

- Changed Get Image Size action to return pixel size.
- Fixed a possible race/crash condition writing to the Last Executed dictionary.
- Possibly fixed a crash related to wireless network triggers.
- Worked around a crashing bug in the system related to the Select Menu Item editor.
- Application popup menu for More re-opens the popup menu.
- Tweaks to the Duplicate Palette hot key handling.
- Alert title was not text token processed.
- Prompt For User Input after Trigger Macro By Name did not have focus.
- Inverted scroll wheel in icon grids to match system behaviour.
- Fixed an issue where the front window might not be restored to focus after closing a clipboard switcher.
- Favorites Category actions is now sorted alphabetically.
- Improved behaviour when launching applications with always hide others.

Changes for 6.1

- Added an Set Safari/Google Chrome Checkbox action.
- Added an Set Safari/Google Chrome Radio Button action.
- Added an action to pop open the Keyboard Maestro status menu.
- Added an action to suspend the login and display the Login Window.
- Conceal probable passwords in the clipboard history.
- Added Cancel Just This Macro (subroutine) action.
- Added Break From Loop action.
- Added a Log action which logs to the Engine.log file.
- Added new Hidden type to plugin actions, so you can easily pass tokenised text.
- Added Acorn-style Feedback panel.
- Prompt For User Input label trims prefixes before double underscore (eg myMacro_Text).
- Improved Accessibility detection and behaviour under Mavericks.
- Read the plain text version of the clipboard in the text tokens when only plain text is desired.

Bug fixes for 6.1

- Chrome Actions had Safari Icons.
- Number Pad numbers were off by one in conflict palette selection.
- Variables preference pane allows deleting of Password variables.
- Plug In actions did not display details in non-edit mode.
- Fixed non-edit display of wireless triggers.
- Fixed non-edit display of show menu action.
- Fixed non-edit display of Set Variable to Calculation to show format.
- Added Disabled on This Mac to non-edit display of macro groups.
- Adjusted ICUDateTimeFor token to be relative to unix time.
- Adjusted clipboard reading code to deal with Safari and its wonky web archives. Again.
- Avoid reading clipboard contents with transient flavour markers.
- Allow the For Each Files collection to scan the root (/) folder.
- The Disabled on this Mac checkbox did not appear until after restarting the editor.
- Fixed the Wrap filter
- Fixed a bug that could affect typing if you changed keyboard layouts.
- Made sure Quit Engine on Editor Quit was not left enabled from MAS version.

Changes for 6.0.1

- Changed re-import plug in action to replace existing plug in action.
- Conflict Palette Option-Number now edits the selected macro.
- Corrected Tutorial highlighting on retina Macs.
- Trigger by Name Command-Option-Number (and Option-Return/Enter) now edits the selected macro.
- Spell Checking is off by default in all text fields.
- Resize Plug In Action Popup Menus to fit their contents.
- Fixed import plug in action to create Keyboard Maestro Actions folder.
- Corrected Universal Access Warning dialog. Sigh.
- Volume Mounted trigger did not display popup menu as "unmounted".
- XML code for Volume Mounted and MIDI trigger were reversed.
- Conflict palette could erroneously highlight numbers.
- Resolved Trigger by Name weirdness when keyboard preference tabs through "All Controls".
- Trigger by Name will not restore items after all items are filtered away.
- Corrected Set Find Pasteboard typo.
- Macro Group Palettes appear in all spaces.
- Fixed a memory leak in the Window Switcher.
- Corrected tooltip on Set Action Timeout button.
- Corrected non-edit display of Move and Resize window.
- Corrected non-edit display of Manipulate Window.
- Avoid reading image files with @2x in their name at retina resolution half size.
- Correct Find Image display location when second monitor is on left.
- Corrected sporadic inverted icons in Application Switcher.
- Fixed Display Status Menu display when Alphabetical was selected.
- Accept documents dropped on editor at launch.

- Use mdfind to scan for applications in the Applications folder (and subfolders).
- Reduced the rate at which Wait For Web Browser queries the readystate.

Major Changes for 6.0

- Requires Mac OS X 10.8.
- Macro Syncing – share your macros across multiple Macs.
- [Plug In Actions](#) – use or write new actions.
- Full support for Styled Text and Named Clipboards.
- Support for controlling Safari and Google Chrome web pages.
- Trigger Macros by Name – trigger any active macro from a single key.
- Fantastic new icons from [Iconaholic](#).
- Customizable macro icons and a new [Icon Chooser](#) and creator.
- [Macro Debugger](#) allows stepping through macros.
- Retina Happy Graphics.

New Triggers

- USB Device is attached or detached.
- Wireless Network is connected or disconnected.
- Volume is mounted or unmounted.
- At Engine Launch.

New Actions

- Plug In Actions – write your own!
- Show a Menu.
- Search Variable or (Named) Clipboard and capture components of regular expressions.
- Get Substring from Variable or (Named) Clipboard.
- Activate macro groups including showing as a palette.
- Copy (Named) Clipboard to (Named) Clipboard.
- Display notification in the Mountain Lion [Notification Center](#).
- Set Variable to Keychain Password and Set Keychain Password to Text.
- Stop Screen Saver.
- Sleep and Wake Screens.
- Set Network Location.
- Set Find Pasteboard.
- Lots of new Debugger actions:
 - Start or Finish Debugging.
 - Control whether new macros start Paused or Running.
 - Breakpoint or Continue this macro or all macros.
 - Breakpoint, Continue, Step Into, Out or Over other macros.
- Toggle Global Macro Palette.

Clipboards & Styled Text

- Apply Styles (Font, Colors, Underlines, etc) to (Named) Clipboard.
- Search & Replace Clipboard preserves styles.
- Filter Clipboard preserves styles.
- Text Tokens preserves styles (particularly the clipboard tokens).
- Named Clipboards preference pane allows editing styled text or setting images.
- Extend Set (Named) Clipboard to Text to support styled text.
- Extend Insert Text to support pasting styled text.
- Extend Write File to support writing styled text.
- Extend Display Text to support styled text.
- Extend Composite Image to support styled text.
- Extend Set Clipboard to Text to allow specifying Named Clipboards.
- Extend Set Clipboard to Past Clipboard to allow specifying Named Clipboards.
- Extend Search & Replace Clipboard to allow specifying Named Clipboards.
- Extend Filter Clipboard to allow specifying Named Clipboards.
- Extend Apply Text Factory to allow specifying Named Clipboards.

Safari and Google Chrome Actions

- New Window with URL.
- New Tab with URL.
- Next Tab, Previous Tab, Select Tab.
- Set URL, Set Title.
- Wait to Finish Loading.
- Click Link.
- Get/Set Fields.
- Submit or Reset a Form.

- Focus or Select a Field.
- Execute JavaScript.
- New text tokens `%SafariTitle%`, `%SafariURL%`, `%SafariReadyState%`, `%SafariField%`, `%SafariJavaScript%`.
- New text tokens `%ChromeTitle%`, `%ChromeURL%`, `%ChromeReadyState%`, `%ChromeField%`, `%ChromeJavaScript%`.
- New functions `SAFARITABINDEX()`, `SAFARITABCOUNT()`, `SAFARIISCOMPLETE()`.
- New functions `CHROMETABINDEX()`, `CHROMETABCOUNT()`, `CHROMEISCOMPLETE()`.

Preferences

- Extend Named Clipboards preference pane to allow editing.
- Added Variables pane to Preferences, including editing.
- Added selectable and contributable status menu icons.

Editor & UI

- Added Sort by Macro Modification.
- Added Sort by Macro Execution.
- Macros and Macro Groups can now have custom icons.
- Status Menu lists and allows you to cancel running macros.
- Added Duplicate command for Macro Groups.
- Detect and report when the system is stuck in Secure Input mode.
- Added a Set Action Timeout button at the bottom of the macro editor.
- Copy as Image and Copy as Text for macros and actions (and contextual menu).
- Enhanced the non-edit display to show full details of actions.
- Enhance the conflict palette to allow selecting with letter keys.
- Added ISO Section, JIS Yen, Underscore, Key Pad Comma, Eisui and Kana to hot key popup menu.
- Hold down shift while pasting from the clipboard switches to paste as plain text.
- Don't expand calculation fields just because they are negative.
- Option click on the Macro Group Palette Keyboard Maestro icon to edit the macro group.
- Adjusted the size of application icons in menus to 16x16.
- Improved the look of the macro group activation toggle display.
- Left/Right Arrows disclose/close actions. Option-Left/Right applies to all actions in that list.
- Remember the location of the conflict palette.
- Recorded clicks can be adjusted to any corner of the main screen or front window after recording.
- Added recording count down before starting recording.
- Added a Pause button to the recording window.
- Added an Insert 1 Second Pause button to the recording window.
- Record mouse drags.
- Record mouse right, centre and other button clicks.
- Added "Or by Ruby script" to script trigger examples.
- Added "Or by script" to Quick Macro action.
- Added Expand/Collapse Action and Expand/Collapse All Actions menu items.
- Remember activation of macro group palettes across changes of applications.
- Changed the View menu Edit item to Start/Stop Editing Macros.

Triggers

- Allow non-ASCII characters in typed string triggers.
- Command or Control keys, or Shift-Space cancels a typed string trigger.
- Allow delete and correct in typed string triggers.
- Allow While Logged in Trigger to trigger ever second (previously the minimum was 3 seconds).

Actions

- Better text insertion of non-ASCII characters with dead keys.
- Extend Resize Image to support Resize to Fit option.
- Enhanced Manipulate Window action to allow specifying an application.
- Extended Execute Script to allow storing the output to (Named) Clipboard.
- Extended Set Variable to Calculation to allow custom number formatting.
- Enhanced Screen Image condition to allow "contains uniquely" and "does not contain uniquely".
- Enhanced performance of finding an image on the screen by 2 to 20 times.
- Added ability to visually see images found on the screen.
- Added option in Activate Application to control whether all windows are activated.
- Add Move & Resize option to Manipulate Window.
- Add defaults options to Move & Resize Window for Left/Right column, etc.
- Added default text setting for Search Web action.
- If Prompt For User Input field starts with a I then token expansion happens before separator parsing.
- Support text token expansion in iTunes and QuickTime Player controls.
- Added Minimize and Unminimize to Manipulate Window.
- Added Select Menu Result, Mouse Click Result and Press Button Result variables to report results.

Text Tokens

- New text token `%l%` allows you to position the cursor after insertion.
- New text token `%lCUDateTimeFor%`.
- New text token `%WirelessNetwork%`.
- New text token `%MacUUID%` to identify this Mac.
- Added Main,Second,Third,Internal,External as special screen indexes for `%Screen%`.
- Support adding months or years in the ICUDateTimePlus/Minus token.

Calculation Functions

- Added `Main`, `Second`, `Third`, `Internal`, `External` as special screen indexes for `SCREEN()`.
- Added `.` notation for accessing point/rectangle coordinates (eg `Screen1.width`).
- Added Context Sensitive Calculation Functions:
 - `IMAGE(Width|Height)` – the action image size.
 - `SOURCEIMAGE(Width|Height)` – the action source image size.
 - `WINDOW(Left|Right|Top|Bottom|Width|Height|MidX|MidY)` – the action window coordinates.
 - `LENGTH()` – the action text length.
 - `FONTSIZE()` – the original font size during the Apply Style action.
- Added an optional percentage offset to the coordinate functions, eg `SCREEN(External,Left,52%)`.
- Added `CLIPBOARDSEED()` function which changes when the clipboard does.
- Allow Em or En dash or Unicode Minus symbol as synonyms for minus in calculations.

Conditions

- Added USB Device Condition to test for the existence of specific devices.
- Added Wireless Network Condition to test for connections to specific networks.
- Added Text Condition.
- Enhanced Menu condition to test for marked (checked) menus.
- Enhanced Button condition to test for checkbox states.
- Enhanced Front Window condition to test non-front applications.

Collections

- Added the ability to count downwards in the Range collection.

In Detail

- Removed the ability of the AppleScript process tokens command to leak Password/PW variables.
- Added `com.adobe.illustrator.hfs` to the excluded clipboard flavour list.
- Removed default 0.01 delay between adding keys to the event queue.
- Reduced the overhead of Pause Until.
- Added option to disable a macro group on just this Mac.
- Added "reload" AppleScript command to the editor.
- Consider a focussed sheet to be the front window for window indexing purposes.

Technical

- Requires Mac OS X 10.8+
- 64-bit only with Automatic Reference Counting.
- Web Server is built into Keyboard Maestro Engine.

Bug Fixes

- Capslock disabled the Command-Tab Keyboard Maestro application switcher.
- Worked around an issue where Safari copies non-text data.
- Move Mouse, with Restore Mouse Location checked but hidden, does nothing.
- Added code to remove "safe save" .dat files from preferences folder.
- Fixed issue if Application Switcher is triggered with just the shift key as a modifier.
- Fixed Switch to Next Application erroneous timeout abort.
- Fixed the power calculation (x^y).
- Fixed the ternary calculation ($x?y:z$).
- Fixed sorting characters showing up in macro group palette titles.
- Fixed a potential crash involving windows without titles.
- Fixed finding an image on the screen on Retina screens.

Download

[Download Keyboard Maestro](#) now to try all these great capabilities. Or keep reading for even more details

about the [Features](#) of Keyboard Maestro.

Alternatively, you can [contact us](#) if you have a question about whether Keyboard Maestro can solve your automation needs. We want all our customers to be satisfied, so we are happy to help you understand how Keyboard Maestro can achieve your automation goals.

Features

Keyboard Maestro is a productivity enhancer with several main functions. It allows you to:

- record and design your own macro shortcuts and activate them at any time.
- work with clipboard history using [Clipboard History Switcher](#).
- work with an unlimited number of saved clipboards using [Clipboard Switcher](#).
- navigate through running applications with [Application Switcher](#) and open windows with [Window Switcher](#).

[Macro Groups](#) allow you to organize your macros. Think of them as folders of macros. Each Macro Group controls when the macros it contains are active. A [Macro](#) is made of two parts: a set of [Triggers](#) you choose to determine when the macro is executed and a list of [Actions](#) that define what the macro does when it is executed.

By creating macros, you can customize your Mac to suit your use, streamline tedious tasks, and remove opportunities for mistakes by automating repetitive jobs. Make a stubbornly difficult applications behave the way you want them to; press a key and have the computer do the next minute worth of tedious tasks for you; type a few characters and have a page full of boilerplate text appear; and so much more. Soon you'll wonder how you could have used your Mac without Keyboard Maestro.

Here are some of the primary features of Keyboard Maestro.

General

- Sync your macros across multiple Macs using DropBox or other file sharing systems.
- Trigger macros by name.
- Use the fantastic status menu icons from [Iconaholic](#).
- [Download](#) or make your own [Status Menu Icons](#).
- Customize your macro icons with the [Icon Chooser](#) and creator.
- Full retina display support.
- Use the included [Macro Debugger](#) for detailed control of your macros in action.

Macro Groups

- Create Macro Groups, which contain [Macros](#) and control when they are active.
- Macro Groups can be restricted to or excluded from specific applications.
- Macro Groups can be activated or deactivated with hot keys, or via the status menu or global macro palette.
- Macro Groups can display a palette of contained macros.
- Macro Groups can be enabled or disabled.
- Customize the Macro Group icon by pasting an icon, or using the [Icon Chooser](#) and creator.
- If you are syncing your macros, Macro Groups can be disabled on specific Macs.

Triggers

- [Macros](#) can be triggered by one or more [Macro Triggers](#) using any number of the following:
 - [Hot Key](#) – press, hold or release a key.
 - [Typed String](#) – type a string of keys.
 - [Applications](#) – on launch, quit, activate, deactivate, or periodically while an application is running or active.
 - [Login](#) – when you log in to your Mac.
 - [Engine Launch](#) – when the Keyboard Maestro engine launches.
 - [System Wake](#) – when the system wakes from sleep.
 - [Time](#) – at a particular time of day.
 - [While Logged In](#) – periodically while logged in.
 - [Macro Palette](#) – with a click on a context sensitive [Macro Palette](#).
 - [Status Menu](#) – by selecting from a global system status menu.
 - [Public Web](#) – over the Internet, explicitly to the public, or via authenticated log in.
 - [Mounted Volume](#) – when a volume is mounted or unmounted.
 - [USB Device](#) – when a USB device is attached or detached.
 - [Wireless Network](#) – when your Mac connects or disconnects to/from a wireless network.
 - [Device Key](#) – when any HID (Human Interface Device) device key is pressed, held down or released.
 - [MIDI Note](#) – when a MIDI note is pressed or released.
 - [iPhone](#) – from your iPhone, iPod touch or iPad.

- By Script – from an AppleScript or a shell script.

Actions

- You can create Macro Actions manually or by recording them.
- You can download or write your own Plug In Actions.
- Macros can execute a sequence of one or more Macro Actions including:
 - Plug In Third Party Actions
 - Application Control Actions
 - Activate Last, Next or a specific application.
 - Bring the current application's windows to the front.
 - Quit All, Other, the current or a specific application.
 - Hide All, Other, the current or a specific application.
 - Show All or a specific application.
 - Clipboard Actions
 - Simulate Cut, Copy or Paste.
 - Set the system clipboard to specific styled tokenized text.
 - Set the system clipboard to a past copy of the clipboard.
 - Set the system clipboard to a variable.
 - Set the system clipboard to a Named Clipboard.
 - Delete the current or a past clipboard entry from the clipboard history.
 - Cut, Copy or Paste to/from a permanent Named Clipboard.
 - Set a Named Clipboard to the system clipboard.
 - Copy a Named Clipboard to another Named Clipboard.
 - Apply styles (font, colors, underlines, etc) to a (portion of) the a clipboard.
 - Apply a BBEdit Text Factory to the current clipboard.
 - Apply various Filters to the contents of the current clipboard.
 - Search and Replace a clipboard.
 - Search a clipboard and extract information.
 - Get a substring of a clipboard.
 - Display a clipboard.
 - Control the Flow and Behavior of a Macro
 - Pause for a number of seconds (possibly calculated).
 - Pause Until conditions are met.
 - Execute a list of actions until conditions are met.
 - Execute a list of actions while conditions are met.
 - Repeat a list of actions a number (may be a calculation) of times.
 - If conditions are met, execute a list of actions, otherwise execute another list.
 - Execute another macro (like a subroutine).
 - Cancel all, other, or this macro.
 - Debugger Actions
 - Start, finish, or toggle debugging.
 - Have new macros begin paused, or run until paused.
 - Breakpoint this, all, or other macros.
 - Step over, into, or out of macros.
 - Continue this or other macros.
 - Note: debugging actions are not paused by the debugger.
 - Mail Actions
 - Send or create a mail message.
 - Set a status flag on the currently selected mail message.
 - Execution Actions
 - Execute an AppleScript and optionally display or store the results.
 - Execute a shell script and optionally display or store the results.
 - Execute an Automator workflow.
 - Execute a JavaScript in the front Safari or Google Chrome window.
 - Execute another macros (like a subroutine).
 - File Actions
 - Reveal a File.
 - Move, rename, copy, duplicate, trash or delete a file.
 - Read and write files (text, styled text or images).
 - Append text to a file.
 - Create a new folder.
 - Read or write a file attribute (eg modification date).
 - Read, set, add, remove or toggle Mavericks Tags.
 - Safari and Google Chrome Actions
 - Create a new window or tab.
 - Move to the next, previous or a specific tab.
 - Wait for the page to finish loading.
 - Set the page URL or title.
 - Focus or select a specific field.
 - Read or write a specific field.
 - Submit or reset a form.
 - Execute a JavaScript in the front window.
 - Image Actions
 - Screen Capture an image.
 - Find an image on the screen.

- Read and write an image file.
- Create a new image.
- Flip, rotate, resize, trim or crop an image.
- Composite a clipboard or styled text onto an image.
- Draw a shape onto an image.
- Display an image.
- Get an image size.
- Interface Control Actions
 - Manipulate a window – resize, move, center, close, zoom, minimize, bring to front.
 - Move or click the mouse with modifiers.
 - Select or show a specific menu.
 - Press a button with a specific name.
 - Type a keystroke.
 - Simulate moving the scroll wheel.
 - Use a variable to set the mouse, window, or application.
- iTunes Actions
 - Play a specific a specific song, a random song or a specific Playlist.
 - Play, pause or stop.
 - Rewind or fast forward.
 - Go to the next or previous track.
 - Raise, lower or set the volume to a specific level.
 - Raise, lower or set the rating of the current song.
- Keyboard Maestro Actions
 - Record a quick macro without launching Keyboard Maestro.
 - Trigger a macro by name.
 - Set (or toggle) whether a Macro Group or Macro is enabled.
 - Activate a macro group (optionally with palette) for one or more actions.
 - Cancel this, all, or other macros.
 - Comment (no action, just for helping you document a macro sequence).
 - Show, hide or toggle the Global Macro Palette.
- Send a MIDI Note or Control Change.
- Notification Actions
 - Display a **Notification Center** notification.
 - Display a growl notification.
 - Display text.
 - Display an alert, optionally stopping the macro.
 - Prompt for user input in a variety of forms.
 - Play a system beep.
 - Play a sound.
 - Speak some text.
 - Highlight a location on the screen.
- Open Actions
 - Open a file, folder, or application.
 - Open the current Finder selection.
 - Open a URL.
 - Open a System Preferences pane.
- QuickTime Player Actions
 - Play or pause the current movie.
 - Step backward or forward.
 - Increase, decrease or set the volume.
- Switcher Actions
 - Activate the application launcher.
 - Activate the application switcher.
 - Activate the window switcher.
 - Activate the named clipboard switcher.
 - Activate the clipboard history switcher.
- System Control Actions
 - Sleep, Restart, Shut Down, Fast User Switch or Log Out.
 - Set the system “Find” pasteboard.
 - Open/close the CD tray.
 - Increase, decrease, set or mute/unmute the system volume.
 - Increase or decrease the brightness.
 - Start or stop the screen saver.
 - Wake the screen.
 - Set the system Network Location.
- Text Actions
 - Insert text by typing or pasting with built-in token expansion.
 - Display text with built-in token expansion.
 - Type a keystroke.
 - Set the clipboard to text.
 - Set a variable to text.
 - Apply styles (font, colors, underlines, etc) to a clipboard.
 - Speak text.
- Variable Actions
 - Set a variable to specific **tokenized** text.
 - Set a variable to a calculation.
 - Set a variable to a clipboard.

- Set a variable to a keychain password.
- Set a keychain password to a variable or text.
- Apply various [Filters](#) to the contents of a variable.
- Search and Replace a variable.
- Search a variable and extract information.
- Get a substring of a variable.
- Use a variable to set the mouse, window, or application.
- Prompt for user information to set a variety of variables.
- Web Actions
 - Open a URL.
 - Search the Web.

Application Launcher

- Display a Cover Flow view of available applications for quick selection.
- Type ahead selection including abbreviations (eg "a p" for Adobe Photoshop).

Application and Window Switcher

- Optionally replace the system Command-Tab application switcher.
- Customize the switcher to match the look you want.
- Switch to any application or window with a keystroke.
- Switch to an application and hide all others.
- Easily select the exact application or window you want.
- Launch, hide, quit or force quit any application.
- Close or minimize any window.
- Quit (or force quit) and relaunch applications.
- Get Info or reveal applications.
- Choose the application ordering you want: alphabetically, by last use, or by launch order.
- Sort windows alphabetically or by window depth order.
- Optionally hide other applications.
- Optionally always hide other applications.

Clipboard Switcher

- An unlimited number of [Named Clipboards](#).
- Copy, Cut or Paste to/from [Named Clipboards](#) using a single keystroke.
- Clipboards are saved permanently.

Clipboard History Switcher

- Never lose your clipboard again.
- Browse your past clipboards and paste any previous clipboard item.
- Send clipboard entries to other Macs.
- Clipboards display rich text and images.
- Use Quick Look to view clipboard entries.
- Set clipboard entries as favorites so they are always available in your clipboard history.
- Clipboard History is optionally saved across logins and restarts.

Purchase

Keyboard Maestro is engineered by Stairways Software Pty Ltd and distributed by [FastSpring](#).

Keyboard Maestro is licensed on a per-user basis and individual users may use it on up to five Macs.

New customers can purchase Keyboard Maestro for US\$36 by choosing [Purchase Keyboard Maestro](#) from the [Keyboard Maestro menu](#). Customers with five or more users should [contact us](#) for a volume discount quote.

Keyboard Maestro 6 is a paid upgrade for most users of previous versions. Existing users are eligible for a discount and can purchase an upgrade by choosing [Purchase Keyboard Maestro Upgrade](#) from the [Keyboard Maestro menu](#).

Customers who purchased Keyboard Maestro directly from us after 1 October 2012 have been issued a free upgrade to Keyboard Maestro 6. If you have not received your free license, you can find your free license upgrade at <http://enquiry.stairways.com/>.

Customers who purchased Keyboard Maestro 5 prior to October 2012 can upgrade to Keyboard Maestro 6 for US\$25. Customers who purchased older versions of Keyboard Maestro can upgrade to Keyboard Maestro 6 for US\$25. If you have not received your instructions on how to upgrade, you can find details by looking up your Keyboard Maestro license at <http://enquiry.stairways.com/>.

Customers who have not disabled upgrade emails have been emailed with new license or upgrade instructions as appropriate. If you have not received this email, please [contact us](#) so we can resolve this promptly.

Customers who purchased Keyboard Maestro on the Mac App Store can convert their license to a direct license by option clicking the "Mac App Version" text in the Keyboard Maestro about box and following the instructions and will then be eligible for a paid upgrade to version 6.

It is our informal policy to have a paid major upgrade roughly once every 12 to 24 months. This allows us to have a reasonably consistent revenue stream with which to fund development of Keyboard Maestro.

A fully-functional trial version of Keyboard Maestro is available for download from <http://download.stairways.com/>.

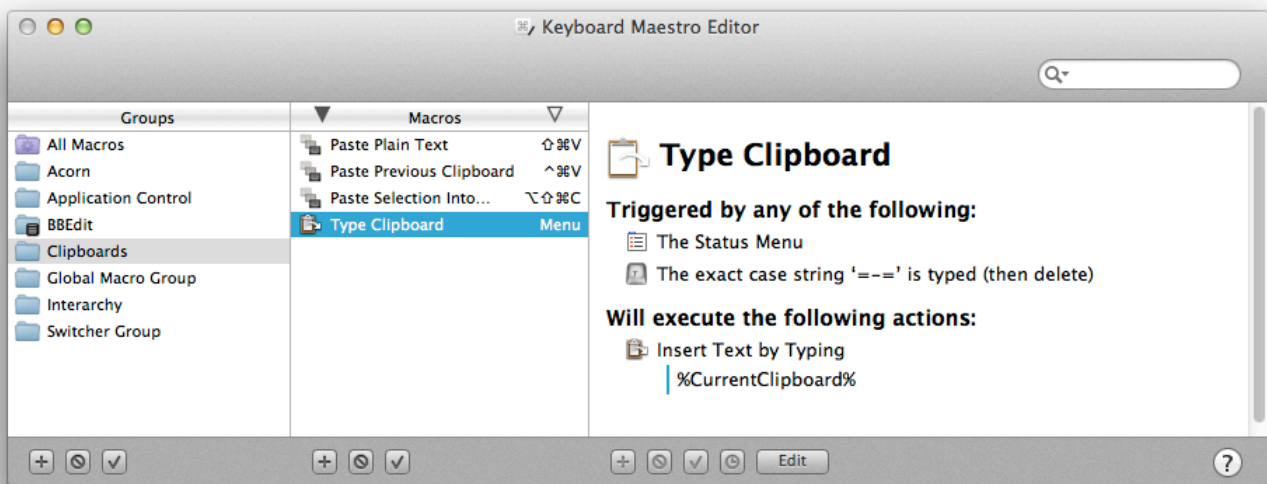
For sales enquires, customer service, technical support, or to contact project management, our current contact information is listed at <http://contact.stairways.com/>.

For more information about anything to do with Keyboard Maestro, visit <http://www.keyboardmaestro.com/>.

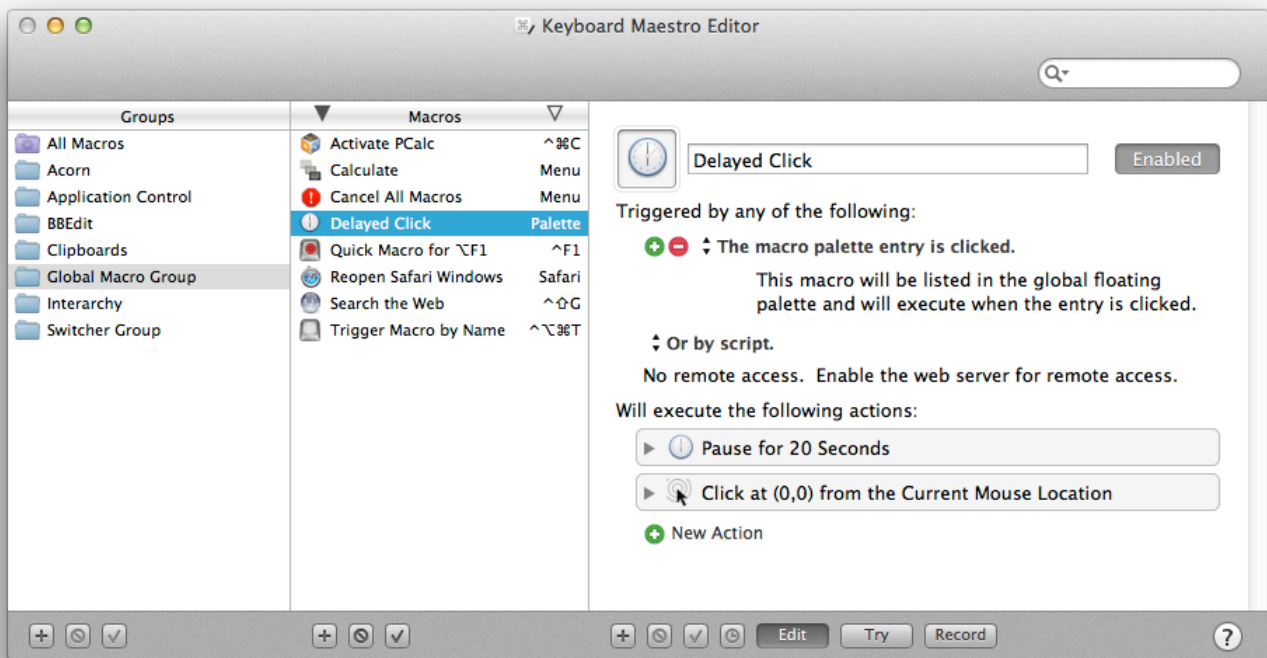
Screenshots

Here is a quick taste of what Keyboard Maestro offers.

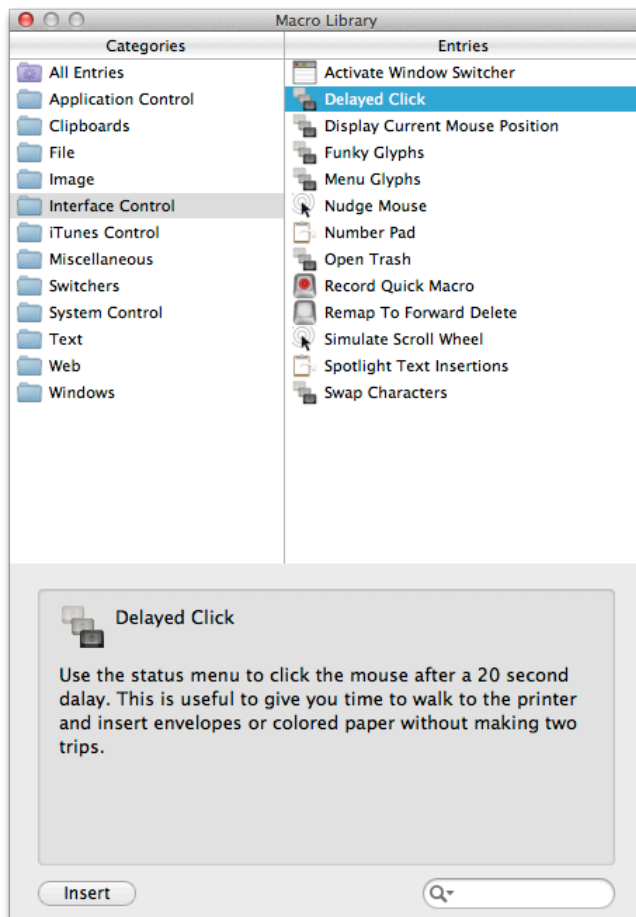
Macros Window



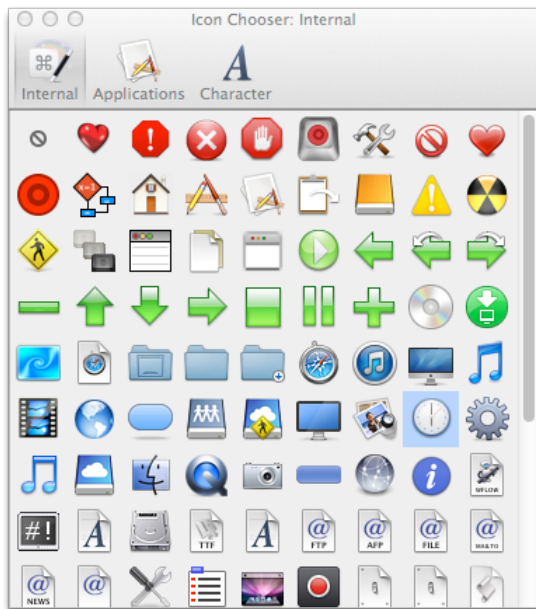
Macro Editor



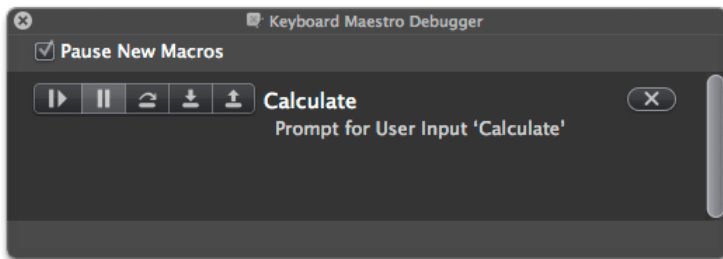
Macro Library



Icon Chooser



Macro Debugger



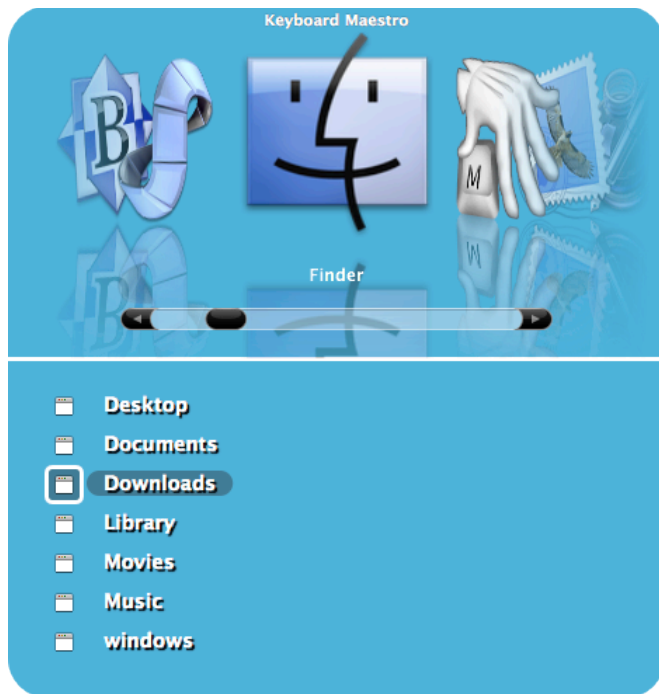
Application Launcher Window



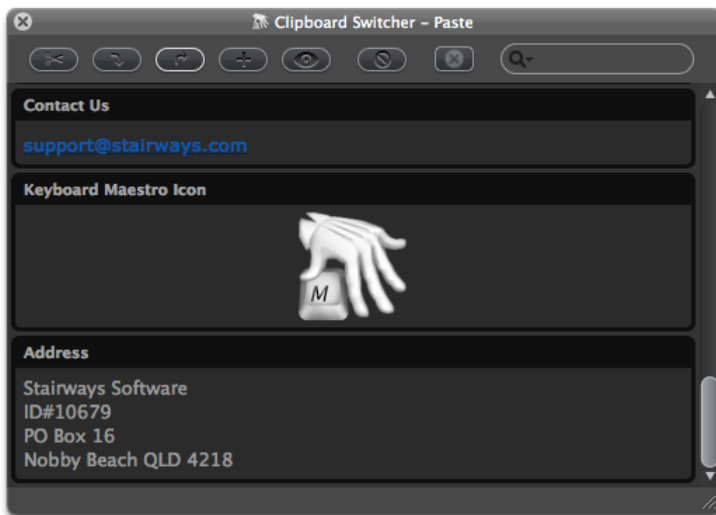
Application Switcher Window



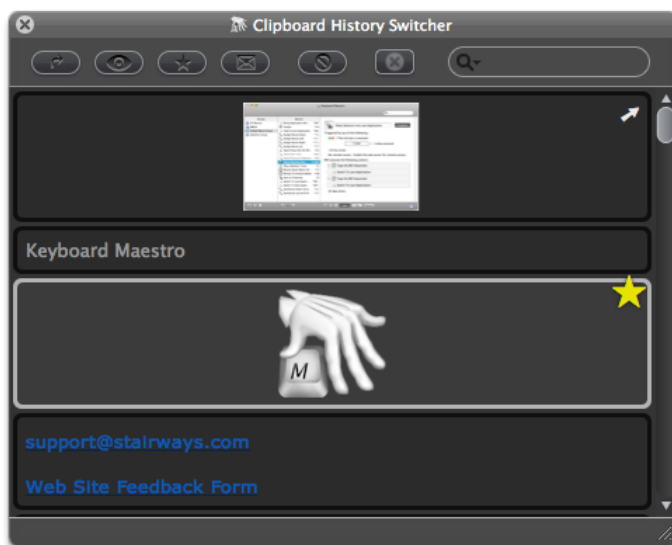
Window Switcher Window



Clipboard Switcher Window



Clipboard History Switcher Window

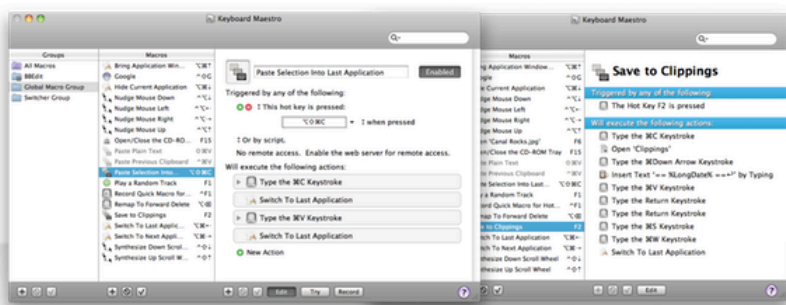


Tour

This demonstration will begin to show you the power and versatility of Keyboard Maestro.

Getting Started

To start, launch Keyboard Maestro. It will initially display the Welcome window.



Welcome to Keyboard Maestro 6

The best way to get to know Keyboard Maestro is to see it in action.

If you're new to Keyboard Maestro, try the Tutorial to see how you can create a simple macro in just a few steps.

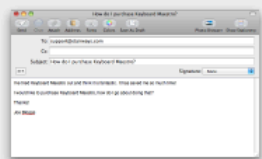
[Start Tutorial](#)

☒ Show this window when Keyboard Maestro opens

Quick Start

Learn about the components of Keyboard Maestro and how they work together.

[Learn more >](#)

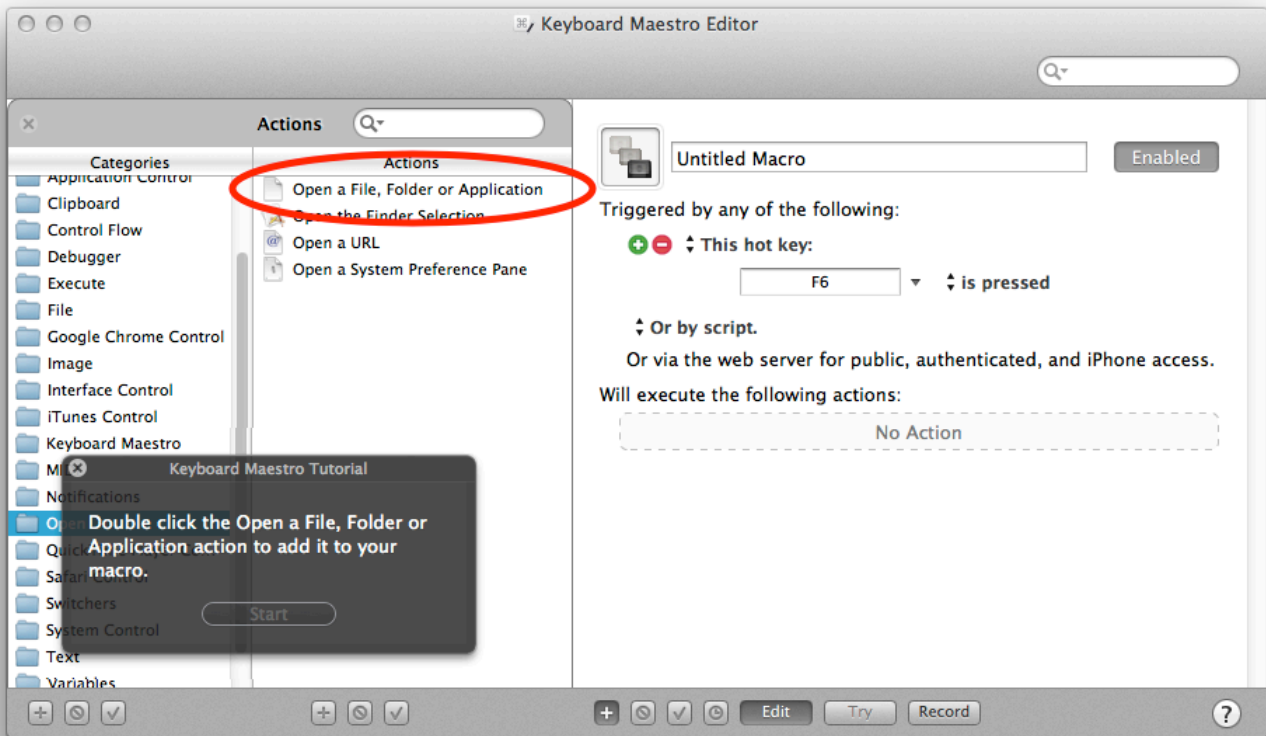


Assistance

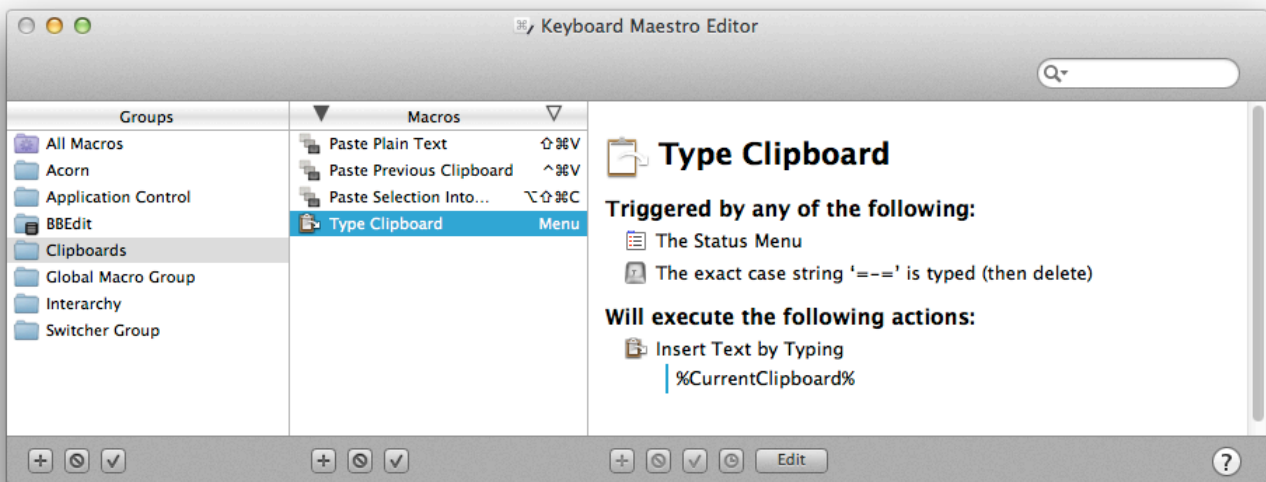
Learn about the user group, documentation, videos, and customer support options.

[Learn more >](#)

If you are new to Keyboard Maestro, start the tutorial and Keyboard Maestro will show you how easy it is to create a macro.



Close the Welcome window to display the Macros window.



You can see some example Macros we have included for you.

Make New Macro

Click the **+** button under the Macro column to add a macro and display the Macro Editor window. We will now design a complex Macro enabling you to save clippings to a text file. First, launch TextEdit and create a new empty document. Save this blank document as Clippings.rtf in your Documents folder.

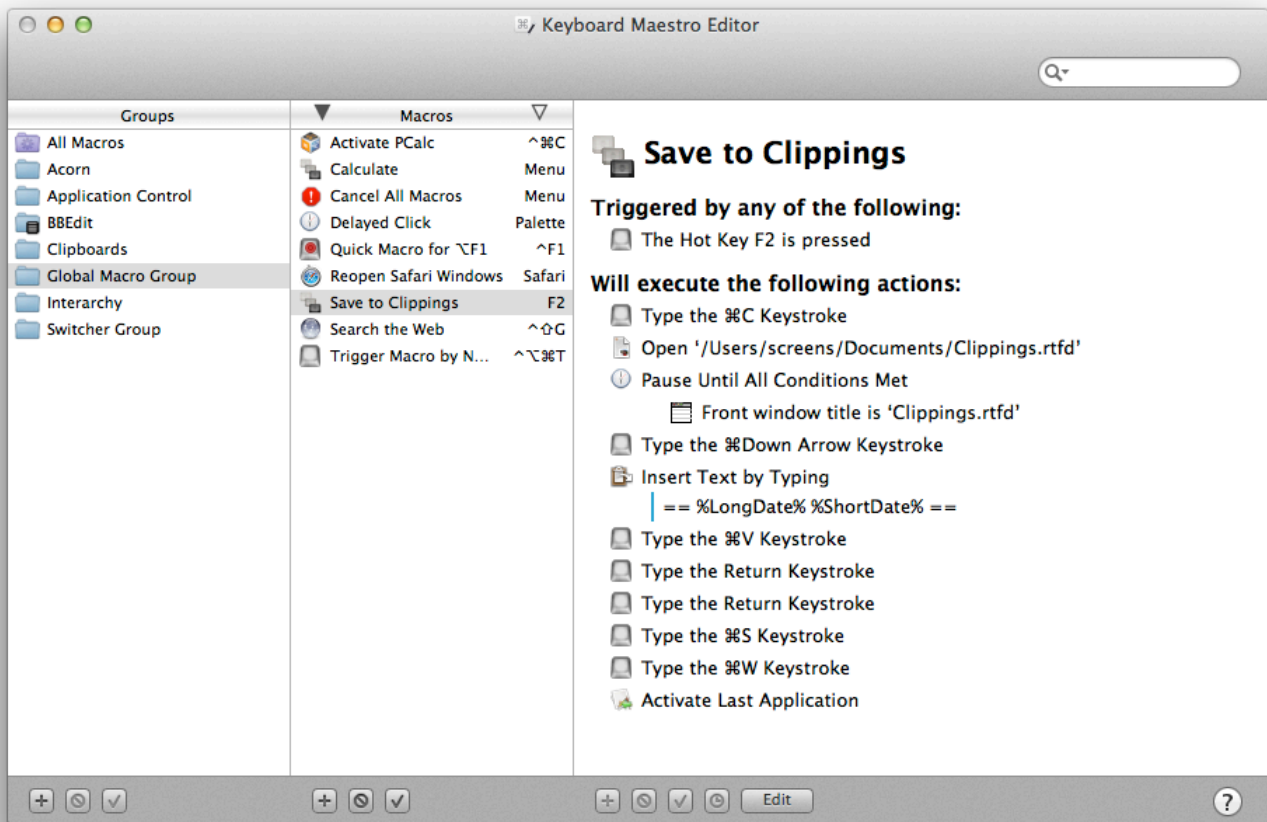
Normally, to add a selection to the Clippings.rtf file, you would have to do all this:

- Press Command-C to copy the selection in an application.
- Go to the Finder, open your Documents folder, then open the Clippings.rtf file.
- Go to the end of the file in TextEdit.
- Press the Return key and type a line of dashes and return to separate the clippings.
- Press Command-V to paste the selection you made before.

- Save the file and close the document.
- Switch back to the application where you originally selected some text or a picture.

That is all very tedious, and probably explains why most people never even bother with such an operation.

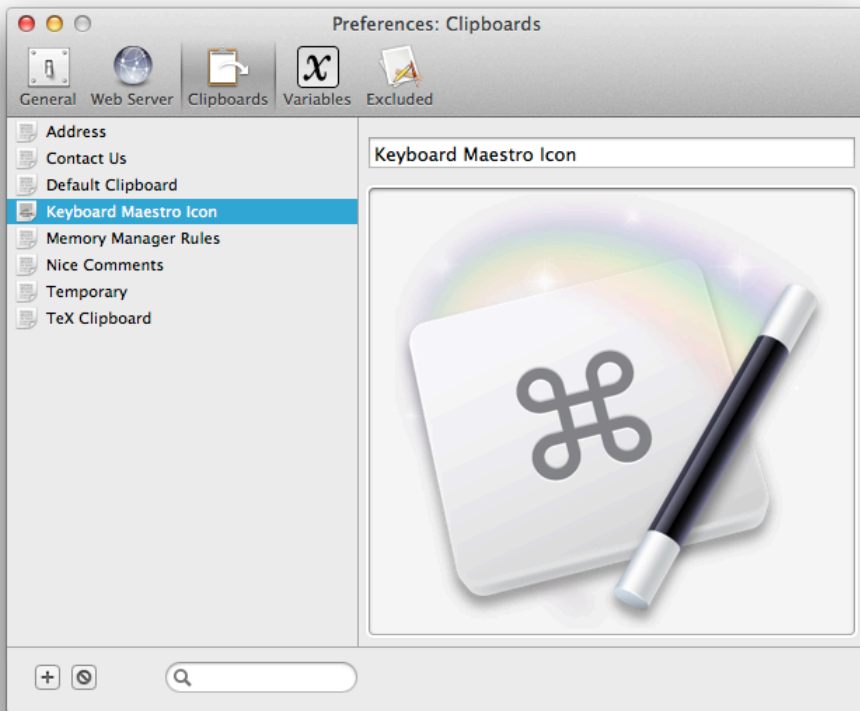
So let's define the whole sequence as a Macro.



Now any time you have some text you want to save, just select it and press F2 (or Fn-F2 depending on the state of the [Use all F1, F2, etc. keys as standard function keys](#) preference in the System Preferences). What used to be too much hassle to bother with is now done in seconds!

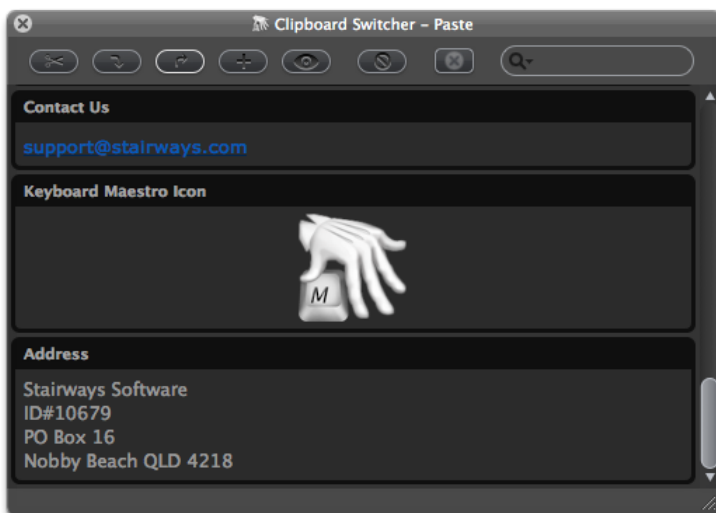
Named Clipboards

Keyboard Maestro lets you create as many named clipboards as you want using the [Clipboards preference pane](#).



Named clipboards let you save frequently used information, like your company logo, timely information like a customer's address, or your address so you never have to type it again. This allows you to paste the saved information whenever you want, wherever you want.

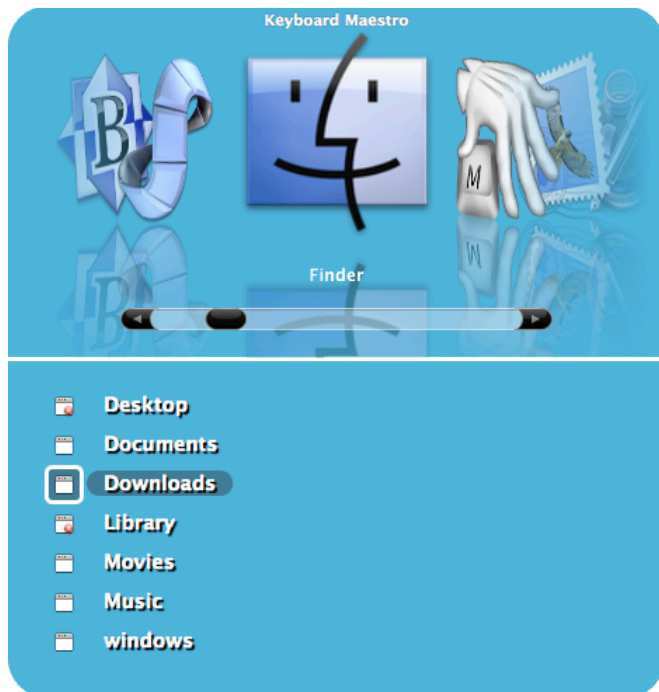
Then you can cut, copy or paste to/from the named clipboards using the defined Hot Keys (by default, Command-Shift-X, C and V respectively, but you can change them, too).



If you have a large screen, or a specific job that needs it, you can have the window stay open (by toggling the Close After Action button) and copy or paste named clipboards easily.

Clipboard History

For less permanent information, Keyboard Maestro automatically remembers your clipboard history, saving a copy of each new clipboard item as you copy it. You can then paste any previous clipboard using the defined Hot Key (by default, Command-Control-Shift-V).



Note the windows set to be closed. This is a very useful way of cleaning up an application with a lot of open windows.

Conduct Your Mac Like a Pro!

This is just a taste of all that Keyboard Maestro can do for you. It's time you started getting the most from your Mac? [Download Keyboard Maestro](#) today and you can be working faster and smarter in no time.

Links

Download Keyboard Maestro from <http://download.stairways.com/>.

Purchase Keyboard Maestro at <http://purchase.stairways.com/>.

Look up your current or previous license status and serial numbers, and get information about discounted upgrades from <http://enquiry.stairways.com/>.

Join the [Keyboard Maestro Forum](#) online community consisting of the developers and Keyboard Maestro users at <http://forum.keyboardmaestro.com/>.

Documentation describing Keyboard Maestro is available at <http://documentation.keyboardmaestro.com/>.

A wiki containing additional information, macros and other resources for Keyboard Maestro is available at <http://wiki.keyboardmaestro.com/>.

For sales enquires, customer service, technical support, or to contact project management, our current contact information is listed at <http://contact.stairways.com/>.

For more information about anything to do with Keyboard Maestro visit <http://www.keyboardmaestro.com/>.

Quick Start

Keyboard Maestro is easy to use once you understand the way the Editor and Engine, [Macro Groups](#) and Macros, Triggers and Actions work together.

The Keyboard Maestro application is the editor. It lets you create and modify macros and configure preferences. You use it only when you want to make changes and then you quit it. It does not always need to be running. Whenever you launch Keyboard Maestro, it also launches the [Keyboard Maestro Engine](#) which continues running until you log out (you can have the [Keyboard Maestro Engine](#) launched automatically when you login by enabling the "Launch Engine at Login" preference in the [General preference pane](#)).



The [Keyboard Maestro Engine](#) is a background only application that enables all of Keyboard Maestro's features. It responds to your [Hot Key](#) presses, watches the time, tracks applications and maintains your clipboard history, handles remote web, iPhone requests and receiving clipboards, and, of course, executes



your [Macro Actions](#). It should be running at all times, so we recommend you enable the “Launch Engine at Login” preference in the [General preference pane](#).

Keyboard Maestro allow you to organize your Macros into [Macro Groups](#) which are like folders of macros. Each [Macro Group](#) controls when the macros it contains are active. A [Macro Group](#) can target or exclude specific applications, which means the macros it contains will only be active in those desired applications. For example, you can have macros which are active only in Mail.app.

A [Macro Group](#) can also act as a container for specific-use macros which are enabled only after a [Hot Key](#) press. For example, you could create a [Macro Group](#) containing macros that resized or repositioned windows using the arrow keys, but those macros would only be active after the [Hot Key](#) was pressed so that the arrow keys could be used normally at other times.


[Macro Groups](#) can be displayed as palettes, allowing you to create your own custom toolbars.


You create a [Macro Group](#) by clicking the  button at the bottom of the [Macro Groups](#) list. You can disable or enable [Macro Groups](#) by clicking the  button. You can configure a [Macro Group](#) by selecting it and clicking the Edit button, or by double-clicking on it.

Keyboard Maestro's main purpose is to execute Macros. A Macro lives in a [Macro Group](#) and consists of a set of Triggers that determine when the macro is executed, together with a list of Actions that define what the macro does when it is executed. You create a Macro by clicking the  button at the bottom of the Macros list. You can disable or enable [Macro Groups](#) by clicking the  button. Keep in mind that a Macro can only be active when the [Macro Group](#) that contains it is active. You can edit a Macro by selecting it and clicking the Edit button.

A Trigger defines when a macro will be executed. There are a variety of Triggers available, the most common is the [Hot Key](#) trigger which executes the macro when a specified [Hot Key](#) is pressed. Similarly, you can use a Typed String trigger to execute a macro when you type some text (for example =addr=). Other common triggers are the [Macro Palette](#) which lets you trigger a macro by clicking on a context (front application) sensitive floating palette of macros and the Status Menu trigger which displays the macro in the Status Menu.

You can also trigger a macro when you login or when your Mac wakes from sleep, at a specific time or on a specific day, when an application launches, activates or quits, by executing a script, or remotely using a web browser or iPhone. And you can trigger a macro when something changes, like a volume being mounted or unmounted, your wireless network connection changing, or a USB device being connected or disconnected.

A Trigger will only execute the macro if the [Macro Group](#) and Macro are enabled and currently active. You create Triggers by creating or editing a Macro and clicking the green  button in the macro detail view.

When a Macro is Triggered it executes a list of Actions. Keyboard Maestro performs each of the Actions in order. There are a wide variety of Actions allowing you to control applications, simulate user interface events like key presses, mouse clicks and menu selections, work with files or images, control your Mac or the clipboard, or display a variety of powerful switchers (Application, Window, Clipboard and Clipboard History Switchers). You can also execute a script (AppleScript, shell Script or Automator Workflow) or even download or create your own custom plug in actions. You create Actions by creating or editing a Macro and clicking the  button to display available actions, or by clicking on the Record button and performing the action while Keyboard Maestro records your actions to your Macro.

By using these six things (Editor and Engine, [Macro Groups](#) and Macros, Triggers and Actions) together, you can dramatically enhance your Mac experience.

- If you want to **make changes**, use the Editor.
- If you want anything to work, make sure the Engine is running.
- If you want to control **when a Macro is active**, configure the [Macro Group](#).
- If you want to control **when a Macro is executed**, configure its Triggers.
- If you want to control **what a Macro does**, configure its Actions.

How do I ...?

- [How do I install Keyboard Maestro?](#)
- [How do I upgrade Keyboard Maestro?](#)
- [How do I purchase Keyboard Maestro?](#)
- [Can I purchase Keyboard Maestro from the Mac App Store?](#)
- [How do I register Keyboard Maestro?](#)
- [How do I get started?](#)
- [How do I create a new Macro?](#)
- [How do I find a Macro I've used or modified recently?](#)
- [How do I cancel a running Macro?](#)
- [How can I get the mouse coordinates on the screen or in a window?](#)
- [How do I insert styled/colored text or images?](#)
- [How do I Insert the current date?](#)

- [How do I get more than one Macro Palette?](#)
- [How do I use a multiple keystroke trigger?](#)
- [How do I configure the Application Switcher?](#)
- [How do I uninstall Keyboard Maestro?](#)
- [How do I revert to a previous version of Keyboard Maestro?](#)

How do I install Keyboard Maestro?

To install Keyboard Maestro, simply copy it to your Mac's [Applications](#) folder (or anywhere you like).

When you launch Keyboard Maestro it launches an invisible “[Keyboard Maestro Engine](#)” that continues to run even after you quit Keyboard Maestro. The engine is the process that enables your [Macros](#), [Application Switcher](#), and [Clipboard Switcher](#) to work. This means that they will continue to work after you quit Keyboard Maestro, as long as the engine is still running.

You can quit or launch the engine manually using the [File menu](#).

You should consider turning on the “Launch Engine at Login” preference in the [General preference pane](#) to ensure all of Keyboard Maestro’s facilities are available to you as soon as you login or startup your Mac.

How do I upgrade Keyboard Maestro?

Keyboard Maestro includes an automatic upgrade mechanism, so to upgrade Keyboard Maestro simply click the [Install Update](#) button when prompted.

To upgrade Keyboard Maestro manually, quit the engine by choosing [Quit Engine](#) from the [File menu](#) and the editor, and replace the Keyboard Maestro application in your [Applications](#) folder with the new one. Finally, launch Keyboard Maestro to restart the engine.

Keyboard Maestro will automatically import your previous version macros, clipboards and preferences. Your old macros will be saved in the [~/Library/Preferences/Keyboard Maestro/Keyboard Maestro Macros Saved Version 5.plist](#) in case you decide not to upgrade to version 6 for any reason.

If you are upgrading directly from a much older version, you will get better results by upgrading to the last of each major version in turn, ie, run 2.1.3, then 3.5, then 4.3.2, then 5.3.2, then the current version. You can download old versions from our [file archive](#).

If you have not done so already, you should consider turning on the “Launch Engine at Login” preference in the [General preference pane](#) to ensure all of Keyboard Maestro’s facilities are available to you as soon as you login or startup your Mac.

How do I purchase Keyboard Maestro?

New customers can purchase Keyboard Maestro for US\$36 by choosing [Purchase Keyboard Maestro](#) from the [Keyboard Maestro menu](#). Customers with five or more users should contact us for a volume discount quote.

You can look up your current or previous license status and serial numbers, and get information about discounted upgrades from <http://enquiry.stairways.com/>.

Thanks for supporting us and enabling us to continue work on Keyboard Maestro.

See also the [Purchase](#) section.

Can I purchase Keyboard Maestro from the Mac App Store?

Keyboard Maestro 6 and future versions of Keyboard Maestro will not be available on the Mac App Store. Apple requires applications on the Mac App Store to be [Sandboxed](#), and workflow applications like Keyboard Maestro cannot be sandboxed so it is excluded from the Mac App Store.

See also the [Purchase](#) section.

How do I register Keyboard Maestro?

When you purchase Keyboard Maestro you will be given a serial number, and will also promptly be emailed your username (email address) and serial number in the “Thanks For Your Purchase” email. Although you can retrieve this information from us at any time in the future, it is a good idea to keep this safe.

If you do not receive your serial number promptly after purchasing, it may be that the email has not reached

you, possibly due to spam filtering on your email service. In this case, try looking up your purchase at <http://enquiry.stairways.com/> (although that will email you your serial number which might again be lost to over-zealous spam filters).

Once you have your username (email address) and serial number, launch Keyboard Maestro and either immediately click the [Use Existing License](#) button or choose [Register Keyboard Maestro](#) from the [Keyboard Maestro menu](#) and then enter the username (email address) exactly as shown and the serial number exactly as shown and click the [OK](#) button. If you have any problems, recheck that the email address and serial number you are entering are exactly as shown (the serial number's email address does not change even if you have changed your email address with us) and also that your license matches the major version number (eg, a version 6 license will work with version 6.x of Keyboard Maestro).

How do I get started?

The first thing to do is to read the [Quick Start](#) and do the tutorial by choosing [Tutorial](#) from the [Help menu](#). You may also want to subscribe to our Getting Started emails (Keyboard Maestro will ask you to subscribe) or you can do that at your [customer database page](#) on our web site.

After that, use your Mac normally and keep an eye out for things you do repetitively. Things like:

- launch or switch to a particular application.
- open a particular document.
- type a specific string of text (eg your name, address, etc).

When you notice something, consider making a Macro to do it and assigning it to a [Hot Key](#) or a [Macro Palette](#) or Status Menu trigger.

Try to be consistent with your [Hot Keys](#), for example you might have a set of applications you open, using a function key for each, and a set of documents you open, using a Control-Function Key combination for each of them, and a bunch of snippets of text you type, using Control-Letter for each of them (the letter could be a mnemonic to help you remember which one is which). Consider putting a sticker on your keyboard across the top of your function keys to help you remember which ones do which function. Remember that you can use the number pad keys as well.

Also, keep in mind common command keys and system defined hot keys and try to avoid conflicting with them.

See also the [Tips](#) section.

How do I create a new Macro?

If you have not done so, use the tutorial by choosing [Tutorial](#) from the [Help menu](#) to lead you through the process of creating a simple macro.

To create a Macro, launch Keyboard Maestro and select the Global Macro group, then click the [+](#) button under the Macros list. Give the Macro a name, add one or more triggers, and one or more actions. The Macro is immediately active.

An easy way to generate macro actions is to turn on recording and proceed to show Keyboard Maestro what you want to do. Then turn recording back off and look through the actions — chances are you will want to delete or adjust some of the recorded actions to make a robust macro, but this will be much quicker than creating each macro action manually.

How do I find a Macro I've used or modified recently?

You can sort the macros by name, by trigger, by last modification or by last execution. So if you launch Keyboard Maestro, select the All Macros pseudo group, and then choose [Sort by Macro Modification](#) from the [View menu](#) or choose [Sort by Macro Execution](#) from the [View menu](#) to sort the recently modified (or executed) macros to the top.

How do I cancel a running Macro?

You can cancel all running macros by holding all the modifiers (Command, Control, Option, Shift) down and clicking on the Keyboard Maestro Status Menu Icon.

You can cancel a specific macro by choosing [Cancel](#) from the [Status Menu menu](#) and selecting the macro. This is also a useful way of seeing what macros are currently running, if any.

You can also see and cancel macros by choosing [Start Debugging](#) from the [Status Menu menu](#).

And finally, you can use the [Cancel All Macros](#), [Cancel Other Macros](#), [Cancel This Macro](#), or [Cancel Just This Macro](#) actions.

How can I get the mouse coordinates on the screen or in a window?

If possible, you should avoid using mouse click actions. They tend to be very fragile, easily broken by subtle changes to the system or applications. And they also require the screen to be in the expected state, so you usually need to add a Pause action before them to ensure the item they are clicking on is where it is expected to be.

That said, there are two easy ways to get the screen coordinates for a mouse click.

The first is to simply record the click. Turn on recording in Keyboard Maestro, go to where you want to be, wait a couple seconds for the screen to be stabilised so the click will be relative to the front window, and click. Turn off recording, delete any extraneously recorded actions and you have your coordinates. Immediately after recording you can adjust the relative corner of the window or screen, and the recorded coordinates will adjust to match.

Alternatively, use the Mac OS X built-in screen capture function to show the coordinates. Press Command-Shift-4 and the mouse will show the screen coordinates. Click and drag relative to the corner of a window to get relative coordinates. Press Escape to cancel the screen capture and enter the coordinates into Keyboard Maestro. Remember that offsets are always to positive to the right and positive down, so if you are making a mouse click up from bottom edge of a window or screen, or left from the right edge of a window or screen, you will need to use negative coordinates.

How do I insert styled/colored text or images?

You can use the [Insert Text](#) action in the [Text category](#) to insert styled text by pasting.

You can insert an image by copying it to the clipboard and then pasting it in using the [Paste action in the Clipboard category](#) (which just types the Command-V keystroke).

You can get your image from a [Named Clipboard](#), or by reading an image file.

Here is how to create a Macro to insert an image when you press a [Hot Key](#).

- Create a macro (see the [How do I create a new Macro?](#) section) with the [Paste from Named Clipboard action in the Clipboard category](#). Select New from the clipboard popup menu, which will create a new named clipboard and show it in the [Clipboards preference pane](#). Name the new clipboard "My Image". Go find your image and copy it, and return to the Preferences window, click on the My Image clipboard on the left side so it is selected, and choose [Paste](#) from the [Edit menu](#) to assign the image to the clipboard.

Name the macro, and assign it whatever trigger you desire.

Now whenever you trigger the macro you just created, your image will be pasted in.

How do I Insert the current date?

You can insert text using the [Insert Text action in the Clipboard category](#), and it processes [Text Tokens](#).

There are some basic date format tokens (such as `%LongDate%`), or you can use the `%ICUDateTime%` [Text Token](#) with any [ICU date format](#).

How do I get more than one Macro Palette?

There are three kinds of palettes in Keyboard Maestro:

- There is one "Global Macro Palette" which includes any active macro that has the Macro Palette trigger. It appears whenever there is any active macro with the Macro Palette trigger. It shrinks to the size of an icon until you hover over it and then it expands to display the currently active macros with Macro Palette triggers. You can show and hide it using specific Macro Palette actions.
- Each Macro Group can be displayed as a palette. The macro group can be global to all applications, or specific to any subset of applications. It can be toggled on and off with a hot key (or a status menu selection or from the Global Macro Palette) or it can be displayed for a single action. Actions can hide or show the macro group palettes.
- When a hot key (or typed string or device key) conflicts (ie, triggers more than one macro), the [Conflict Palette](#) appears which lets you select from the conflicting macros. This can be an easy way to allow a single hot key to offer a multitude of similar actions. You can then use number keys or the mouse to select the desired hot key, or can use other keys to filter the palette until only one macro remains.

So to have more than one macro palette, create a macro group for each desired palette and configure it to show a palette as desired. Put your macros in there. Create as many of these as you like. The macros in

such a macro group are only active while the palette is displayed, so if you only display it occasionally, especially only for one action, then they can have very simple hot keys (like plain letters for example).

You can control the order of macros in a macro palette (or the status menu) by prefixing their name with two characters and a closing parenthesis (eg "01)" - two characters and a closing bracket). The macros will be sorted based on the code, but the code will be stripped off before display in the palette (or status menu).

How do I use a multiple keystroke trigger?

You can assign the same hot key to multiple macros, and Keyboard Maestro will display a palette of the conflicting macros when you press that hot key allowing you to select the desired macro. You can select a macro from the palette using either number keys, or by typing the first distinct character to filter the macros down until only one is left. This is especially useful when you have a variety of similar or related tasks. You can also assign a hot key to a macro group which can activate it for one action (with or without a palette), and the contained macros can have whatever "second" hot key you desire.

But Keyboard Maestro does not directly support assigning a two-keystroke hot key to a trigger. The problem with multiple keystroke triggers like Option-F R is what to do if you type Option-F A? Logic dictates that the Option-F A should go through to the system unimpeded, but Option-F R should be swallowed entirely. But this is impossible. The only way to do it would be to swallow the Option-F key, and then swallow the second key and then resubmit the Option F and the second key unless it matches Option-F R.

However, that is fraught with peril and cannot work robustly in the presence of other applications placing things on the keyboard event queue (or even a sufficiently fast typist).

For example, suppose you quickly typed Option-F A B. Keyboard Maestro would have swallowed the Option F and then the A, and then resubmitted it to the event queue, resulting in the stream of characters B, Option-F, A. There is no way to avoid this race condition, and as such Keyboard Maestro does not support any such mechanism.

As described above, Keyboard Maestro has a variety of ways you can use Option-F as a hot key that allows a second key to be used to select a macro. However in all cases it is clear that the Option-F has been used and there is no concept that the Option-F might come back later to do something else.

How do I configure the Application Switcher?

The Application Switcher (and all the switchers) are activated by macro actions. Keyboard Maestro creates a default "Switcher Group" Macro Group containing several macros, each macro has a hot key trigger and a matching action which activates the appropriate switcher.

So to configure the Application Switcher, launch Keyboard Maestro, select the Switcher Group, and double click the Activate Application Switcher macro. You can then configure the various Application Switcher parameters, such as style and icon size by configuring the Application Switcher action. You can also configure the hot key used to activate the switcher, or disable the switcher.

How do I uninstall Keyboard Maestro?

Launch Keyboard Maestro and ensure the "Launch Engine at Login" preference in the [General preference pane](#) is turned off. Then choose [Quit Engine](#) from the [File menu](#) to quit the engine, and then choose [Quit Keyboard Maestro](#) from the [Keyboard Maestro menu](#) to quit the application.

You can then trash the Keyboard Maestro application from your [Applications](#) folder, as well as the preferences in the [~/Library/Application Support/Keyboard Maestro](#) folder and [~/Library/Preferences/com.stairways.keyboardmaestro.*](#) files and logs in the [~/Library/Logs/Keyboard Maestro](#) folder.

How do I revert to a previous version of Keyboard Maestro?

Launch Keyboard Maestro and ensure the "Launch Engine at Login" preference in the [General preference pane](#) is turned off. Then choose [Quit Engine](#) from the [File menu](#) to quit the engine, and then choose [Quit Keyboard Maestro](#) from the [Keyboard Maestro menu](#) to quit the application. You can then trash the Keyboard Maestro application from your [Applications](#) folder.

For version 2, open the [~/Library/Preferences](#) folder and the folder [~/Library/Preferences/Keyboard Maestro/Saved Version 2](#) folder. Move the files from the latter folder into the former folder. Trash the [~/Library/Preferences/Keyboard Maestro](#) folder. Download Keyboard Maestro 2.1.3, if necessary, from <http://files.stairways.com/keyboardmaestro/keyboardmaestro-213.dmg>. Move Keyboard Maestro 2 to the

Applications folder and launch it. Turn on the “Launch Engine at Login” preference if desired.

For version 3, open the `~/Library/Preferences/Keyboard Maestro` folder. Trash the `Keyboard Maestro Macros.plist` and replace it with the `Keyboard Maestro Macros Saved Version 3.plist`. Download Keyboard Maestro 3.5, if necessary, from <http://files.stairways.com/keyboardmaestro/keyboardmaestro-35.zip>. Move Keyboard Maestro 3 to the Applications folder and launch it. Turn on the “Launch Engine at Login” preference if desired.

For version 4, open the `~/Library/Preferences/Keyboard Maestro` folder. Trash the `Keyboard Maestro Macros.plist` and replace it with the `Keyboard Maestro Macros Saved Version 4.plist`. Download Keyboard Maestro 4.3.2, if necessary, from <http://files.stairways.com/keyboardmaestro/keyboardmaestro-432.zip>. Move Keyboard Maestro 4 to the Applications folder and launch it. Turn on the “Launch Engine at Login” preference if desired.

For version 5, open the `~/Library/Application Support/Keyboard Maestro` folder. Trash the `Keyboard Maestro Macros.plist` and replace it with the `Keyboard Maestro Macros Saved Version 5.plist`. Download Keyboard Maestro 5.3.2, if necessary, from <http://files.stairways.com/keyboardmaestro/keyboardmaestro-532.zip>. Move Keyboard Maestro 5 to the Applications folder and launch it. Turn on the “Launch Engine at Login” preference if desired.

Macro Groups

Keyboard Maestro organizes your macros into Macro Groups which are like folders of macros. Each Macro Group contains a number of macros and controls when those macros are active.

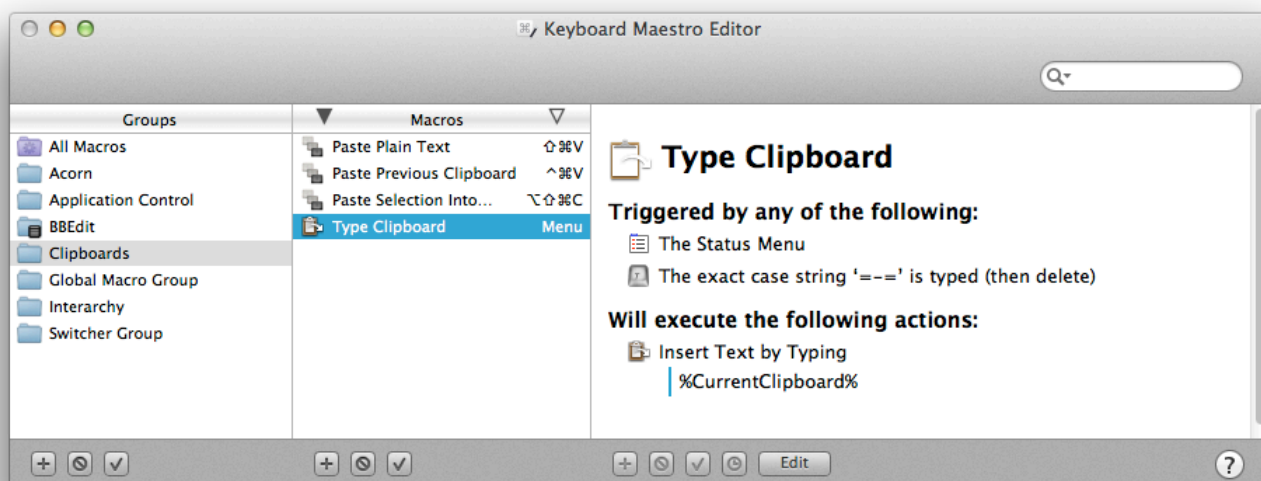
A Macro Group can target or exclude specific applications, which means the macros it contains will only be active in those desired applications. For example, you can have macros which are active only in Mail.app.

A Macro Group can also act as a container for specific-use macros which are enabled only after a Hot Key press or which are displayed as a palette of the macros. For example, you could create a Macro Group containing macros that resized or repositioned windows using the arrow keys, but those macros would only be active after the Hot Key was pressed so that the arrow keys could be used normally at other times.

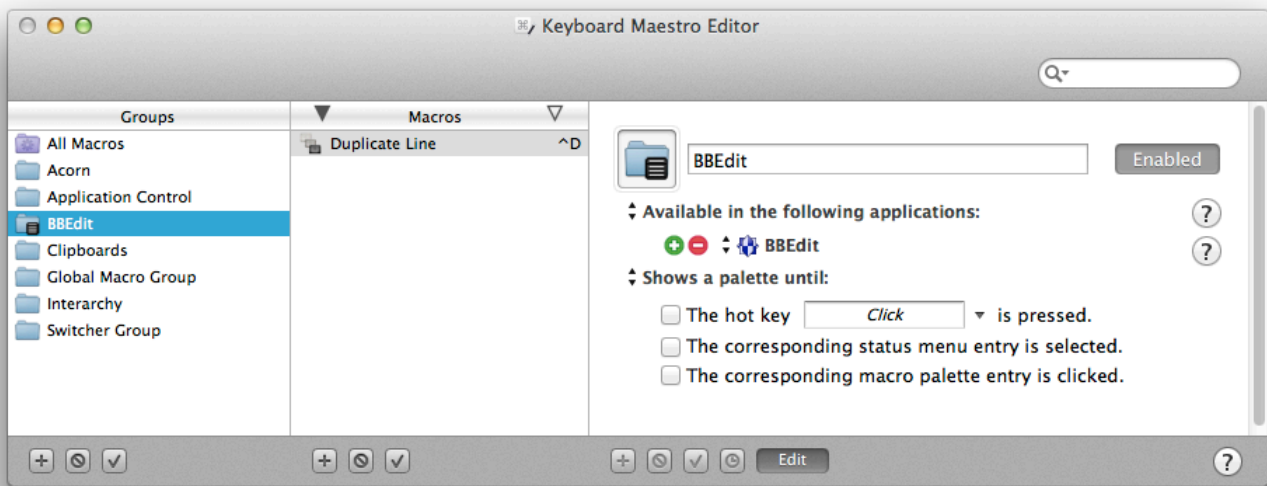
Macro Groups can be displayed as a palette giving you a way to build custom toolbars.

If you are syncing your macros with another Mac, Macro Groups can be disabled specifically on this Mac.

To create a new Macro Group, first launch Keyboard Maestro.



Now click the `+` button below the Macro Groups list.



Enter the name of your new Macro Group.

You can choose to target the macros in your Macro Group at specific applications.

By default, Macro Groups and their Macros are available in all applications, that is they are always ready to be triggered. These are especially useful for Macros that give you instant access to applications or documents, or type in globally applicable text. For example you might have a Macro to launch your email client or word processor, a Macro to open your financial accounts, and a macro to type your name or email address.

To have macros only active in specific applications, set the Macro Group to be “Available in the following applications” and add the desired applications to the list.

For example, you could have macros targeted at:

- Mail.app that insert common text messages.
- BBEdit and Xcode that insert code chunks or duplicate lines or add #include headers.
- Safari that configure windows or enter information.
- Photoshop or Acorn that arrange items or script guides.

To have macros active everywhere **except** specific applications, set the Macro Group to be “Available except in the following applications” and add the desired applications to the list. For example, you could exclude macros from triggering in games. Also, if you have an application that uses lots of function keys for crucial tasks, you could exclude that application to allow you to use the function keys for macros elsewhere without conflicting with that application.

You can also choose to activate the macros manually or display the macros in a floating macro palette. The options are:

- Always activated.
- Activated for one action when:
- Activated/deactivated when:
- Shows a palette for one action when:
- Shows/hides a palette when:
- Shows a palette until:

To have the macros in a Macro Group always ready to be triggered, set the Macro Group to be “Always activated”.

To have macros that are active only immediately after you trigger the Macro Group with no visible palette, set the Macro Group to be activated “Activated for one action when”. The macros in the Macro Group will be enabled when you trigger the Macro Group and will remain enabled until either any macro is triggered or you press any other key. You could use this to create a set of related actions with easily remembered hot keys that will not conflict with normal use because they are not activated until you trigger the group. For example, you could have a group of macros to launch various applications so that Command-Control-L activates the group and then a single letter press launched the application (eg M for Mail, S for Safari, F for Finder).

As with each of the following options, you can trigger the macro group by pressing a Hot Key, by selecting from the Status Menu, or by clicking on the global floating palette.

To have Macros that are remain active after you trigger the Macro Group, set the Macro Group to be “Activated/deactivated when”. The Macros in the Macro Group will be enabled when you trigger the Macro Group and will remain enabled until you dismiss the Macro Group by repeating the trigger. You could use this to create a set of related actions with easily remembered hot keys that will not conflict with normal use because they are not activated until you press the group Hot Key. For example, you could have a group of macros to move and resize windows and have Command-Control-W activate the group. Then a single arrow key press moves the front window. When the window is positioned, press Command-Control-W a second time to disable the macros.

To have Macros that are active and displayed in a macro palette only immediately after you trigger the Macro Group, set the Macro Group to “Shows a palette for one action when”. The Macros in the Macro Group will be displayed in a floating Macro Palette and enabled when you trigger the Macro Group and will remain displayed until either any macro is triggered or you press any other key. You could use this to create a set of related actions that do not even need a Hot Key. For example, you could have a group of macros to launch various applications like Mail, Safari and the Finder so when you press Command-Control-L, a palette of these macros is displayed and a single click on the desired application will launch the application.

To have Macros that are active and displayed in a macro palette after you trigger the Macro Group, set the Macro Group to “Shows/hides a palette when”. The Macros in the Macro Group will be displayed in a floating Macro Palette and enabled when you trigger the Macro Group and will remain displayed until you trigger the Macro Group again. You could use this to create a set of related actions that do not even need a Hot Key. For example, you could have a group of macros to align objects in a CAD application, so when you press Command-Control-A, a palette of these macros is displayed and you can click various alignment options (distribute left-right, align top edges) and then close the palette by pressing Command-Control-A a second time.

To have Macros that are always active and displayed in a palette, set the Macro Group to “Shows a palette until”. The Macros in the Macro Group will be displayed in a floating Macro Palette and enabled. The palette will close when (if) you trigger the Macro Group and will remain closed (and the macros disabled) until you trigger the Macro Group again. You could use this to create a set of related actions that do not even need a Hot Key and that are available in a Macro Palette. You could make the Macro Group available only in a specific application so that it appears only in that application. For example, you could have a group of macros to align objects in a CAD application, and have the Macro Group available (and hence the palette displayed) only in the CAD application.

Even if the macros are displayed in a Palette, the macros can still have Hot Key (or any other kind of) triggers which will be available whenever the macro group is active (in this case, whenever the palette is displayed).

You can disable or enable a macro group by selecting it and clicking the ☒ button at the bottom of the Groups list. If a Macro Group is disabled, all its contained Macros will be disabled. You can disable or enable a Macro Group using the Set Macro Enable action, you can show in a palette or activate a Macro Group using the Show Macro Group or Activate Macro Group actions.

You can change the targeting of an existing group by selecting it in the Groups list and clicking the button, or by double clicking it.

You can disable a Macro Group from AppleScript with:

```
tell application "Keyboard Maestro"
    setMacroEnable "Macro Group Name or UID" with/without enable
end tell
```

You can start editing a Macro or Macro Group from AppleScript with:

```
tell application "Keyboard Maestro"
    editMacro "Macro Group Name or UID"
end tell
```

You can get the list of currently selected macros or macro groups with:

```
tell application "Keyboard Maestro"
    selectedMacro
end tell
```

To delete a Macro Group, select the macro group and then press the Delete key or click the button at the bottom of the Groups list.

You cannot delete or rename or disable the Global Macro Group. If you want to control when some of the macros within the Global Macro Group are active, make a new macro group and drag those macros to it.

Macros

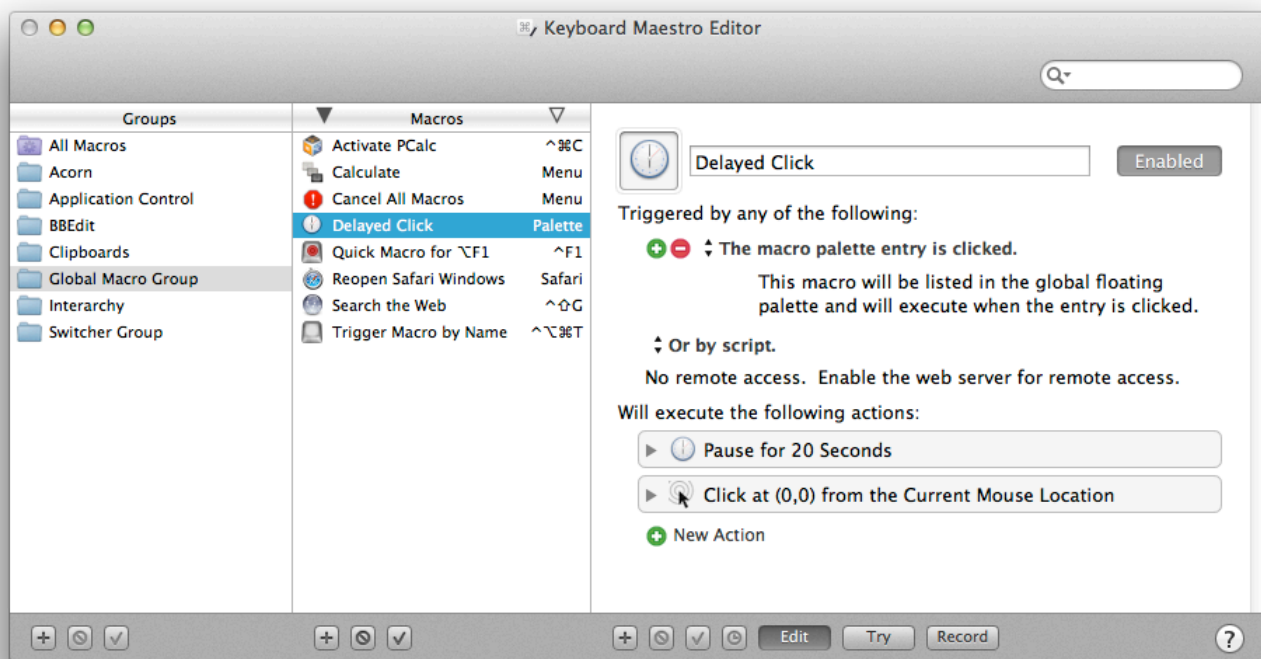
Macros, also known as Shortcuts, are a way of improving your productivity by allowing you to perform

repetitive or frequently required actions more quickly and accurately, tailoring your Mac to your usage patterns.

A macro consists of a set of zero or more possible [Macro Triggers](#) which define when the macro will be executed, together with an ordered list of [Macro Actions](#) to be performed. Sets of [Macros](#) are collected together into [Macro Groups](#).

A typical simple Macro consists of a single [Hot Key](#) trigger, such as Control-A, together with a single action, such as “type my address” .

To create a Macro, first launch Keyboard Maestro, select the desired Macro Group to contain it, and then click the [+](#) button below the Macros list. To edit a Macro, double click it, or select it and press the [Edit](#) button. The [Macro Editor window](#) will be displayed.



Enter the name of your new Macro (you can skip this and it will be named for you based on the action you select).

You can create a new trigger by clicking on the green [+](#) button. There are several [Macro Triggers](#) to choose from, the most common being the [Hot Key](#) which allows you to execute the Macro at the press of a key. You can define several different triggers, and any of the triggers will execute the [Macro Actions](#). You can delete a trigger by clicking the red [-](#) button.

You can add a new action by clicking the [New Action](#) button or the [+](#) button at the bottom of the macro detail view to display the list of actions. There are many [Macro Actions](#) to choose from. The actions you include will be executed in order. You can reorder the actions by dragging them around. You can copy actions by Option-Dragging or by using Copy and Paste. You can delete an action by selecting it and pressing the Delete key. You can enable or disable actions by selecting them and clicking the [✓](#) button at the bottom of the macro detail view. You can set the timeout of an action by clicking the [⌚](#) button at the bottom of the macro detail view

An easy way to generate macro actions is to turn on recording by clicking the [Record](#) button at the bottom of the macro detail view and proceeding to show Keyboard Maestro what you want to do. Then turn recording back off and look through the actions. Chances are you will need to delete or adjust some of the recorded actions to make a robust macro, but this will be much quicker than creating each macro action manually.

You can display more or less detail about some actions by clicking the disclosure triangle. While disclosed, you can try the action immediately by clicking the [Try](#) button at the top right of the action. You can disclose all the actions (in a sublist) simultaneously by option clicking on the disclosure triangle.

You can try the actions immediately by selecting some or all of them and clicking the [Try](#) button at the bottom of the macro detail view.

Macros are continuously saved, so the macro is live as soon as it is created. It will be available immediately

(subject to the restrictions of the [Macro Group](#) it is contained in).

You execute a macro's action sequence by triggering the macro using any of the [Macro Triggers](#) you have defined.

You can also trigger macros by name using the [Trigger Macros by Name](#) action, which in turn can be in a macro and triggered any way you desire.

If the web server is enabled for remote access, you can trigger a macro remotely after logging in using a web browser or the Keyboard Maestro Control iPhone application.

You can also trigger a macro using AppleScript or another scripting language (select the "Or by script" entry to display script code in various languages such as AppleScript or Perl).

Note that the web server needs to be separately enabled in the [Web Server preference pane](#), and all macros are subject to the restrictions of the [Macro Group](#) they are contained in. If the [Macro Group](#) is not enabled and active, the macro will not be available.

You can disable or enable a macro by selecting it and clicking the ☒ button at the bottom of the Macros list.

You can also disable or enable a Macro using the Set Macro Enable action, or from AppleScript with:

```
tell application "Keyboard Maestro"
    setMacroEnable "Macro Name or UID" with/without enable
end tell
```

You can start editing a Macro or Macro Group from AppleScript with:

```
tell application "Keyboard Maestro"
    editMacro "Macro Group Name or UID"
end tell
```

See also the [Macro Groups](#), [Macro Actions](#), [Macro Triggers](#) and [Recording](#) sections.

Macro Triggers

- [Overview](#)
- [Hot Key](#)
- [Typed String](#)
- [Application](#)
- [Login](#)
- [Engine Launch](#)
- [System Wake](#)
- [Time](#)
- [While Logged In](#)
- [Macro Palette](#)
- [Status Menu](#)
- [Public Web](#)
- [Mounted Volume](#)
- [USB Device](#)
- [Wireless Network](#)
- [Device Key](#)
- [MIDI Note](#)
- [By Script](#)

Overview

A Macro is executed when any of its [Macro Triggers](#) is activated. There are several triggers to choose from (detailed below), the most common being a [Hot key](#), that is a Macro is executed in response to a keystroke, usually in conjunction with one or more modifier keys. You can also trigger a macro by typing a string. Or you can display the containing [Macro Group](#) as a floating palette, or execute macros remotely via the built-in web server.

You can also trigger macros by name using the [Trigger Macros by Name](#) action, which in turn can be in a macro and triggered any way you desire.

Hot Key

The most common [Macro Trigger](#) is the traditional [Hot Key](#). You execute a Macro by pressing a keyboard key like a letter, number, symbol or function key, often in combination with one or more modifiers (Shift, Control, Option and/or Command). Almost any key can be a trigger, and keep in mind the number pad counts as different keys to the numeric keys on the main keyboard.

You can type the desired key or key combination in the hot key box, or select a predefined key (holding

down any desired modifiers) from the popup menu to its right. Note that there is a relatively prevalent [third party / system bug](#) that makes the system think it is permanently in a password field, and this will prevent entering a hot key by typing in this manner.

The macro can execute when the hot key is pressed, released or repeat continuously while it is held down. This allows you to do things like have a macro execute when the key is pressed, and then a second macro execute when the key is released, for example to toggle a setting on and then off again.

A common use for [Hot Key](#) triggers is to open applications or documents, insert text templates, or to remap command keys (although you can also remap command keys in the System Preferences Keyboard preference settings).

[Hot Keys](#) suffer from the drawback that you need to remember a cryptic keystroke. This can be mitigated by selecting consistent keystrokes (such as Control-Letter to mean insert text and Control-Option-Letter to mean launch an application). You can also use a tool like [KeyCue](#) to display command keys and macro hot keys.

To further help with this, if multiple macros are executed with the same hot key, the conflicting macros are displayed in a palette allowing you to select the desired macro. You can select a macro from the palette using either number keys, or by typing the first distinct character to filter the macros down until only one is left. You could use this feature to allow a single hot key to do multiple user-selected actions.

Typed String

The Typed String trigger lets you execute a macro in response to a sequence of keys.

Typed String triggers allow you to use more verbose (and hence descriptive) sequences of keys to trigger a macro. Because the keys first go through to the current application, the keys are usually deleted prior to executing the macro, although with this disabled you can use it in an application that largely ignores keys. Triggers can include non-ASCII characters, but you should verify that the deletes work appropriately in these cases.

WARNING: You should generally not use a Typed String trigger that simulates deletes in a non-text typing context, as the simulated delete keystrokes could be destructive.

To avoid macros firing unexpectedly it is a good idea to include a consistent prefix and/or suffix to your strings. For example, to insert your email address, rather than use just "em" (which would fire if you typed "them"), use something like "=em=" which you will not type accidentally.

Keyboard Maestro will accept the trigger even if you use the delete key to correct it (for example, in the case above, if you typed "=en«delete»m=" the trigger will still fire. If you need to type the trigger text without the macro firing, type and delete a Shift-Space in the middle, like "=e«Shift-Space»«delete»m=".

As above, typing Shift-Space will cancel the sequence, as will any control key or command key (eg Command-), any macro execution, switching applications, clicking the mouse, or not typing for more than a few seconds.

You can optionally allow typed string triggers to work regardless of case, regardless of diacriticals, or to remember the case of the typed string and mimic the case in any Insert Text actions. You can limit the typed string trigger to only work after a word break (which means any non-alphanumeric character, or any case that cancels the sequence as described above).

Alternatively, you can use a regular expression to match a trigger. The trigger is matched against the end of the typing sequence, so there is an implicit \z on the end of the regular expression. Note that it would be useless to try to match a word break at the end with \b or some sort of positive or negative lookahead assertion as the typing sequence always ends at the current character. So for example "ell\b" would match "hell" even if the next character you planned on typing was an "o".

You can use the %TriggerValue% [Text Tokens](#) to determine the exact text that was typed.

If multiple typed string triggers match from the end (eg "hello" and "llo"), the longest will be used. If there are multiple matches with the same length (as can occur if you use the case or diacritic options), the conflict palette will be displayed, offering a selection of the matching macros. In this case, if all of the typed string triggers have the "Simulate N deletes before executing" option enabled, the deletes will be simulated immediately and then the conflict palette will be displayed; otherwise the deletes will not be simulated.

Note that the system will not allow Keyboard Maestro to see keys typed in password fields, so Typed String triggers will not fire if you type them in most password fields. You can use Insert Text by Typing to type in to password fields (although this is a serious security concern), but you cannot use Typed String triggers while inside a password field.

Note also that there is a relatively prevalent [third party / system bug](#) that makes the system think it is

permanently in a password filed, and thus will prevent typed string triggers from working.

Application

You can have a Macro execute in response to an application event, such as when the specified application launches, quits, activates or deactivates. You can also have the Macro run periodically while an application is running or while it is at the front.

You could use a trigger like this to simulate workspaces by automatically setting up an application the way you want when you launch it, or you could clean up after an application when you quit.

As with all triggers, the trigger will fire only if the Macro Group that contains it is active, which is based on the current foreground application **before** the specified application launches, or **after** the specified application is deactivated or quits. In practice, this means the Macro Groups that contain this trigger should be targeted at All Applications.

Login

You can use the Login trigger to execute a macro when you login, assuming Keyboard Maestro Engine is launched at login, as it will be if the “Launch Engine at Login” preference in the [General preference pane](#)) is checked.

You could use a trigger like this to set up your Mac environment when you start your Mac.

Engine Launch

You can use the Engine Launch trigger to execute a macro when the Keyboard Maestro engine launches. Typically, this will be when you login, but it could also be at other times if you quit and restart the engine for any reason, or launch the editor or engine at some time after login.

System Wake

You can use the System Wake trigger to execute a macro when your Mac wakes from sleep.

You could use a trigger like this to set up your Mac environment, first determining your location and then taking appropriate action.

Time

You can use the Time trigger to execute a macro at a specific time, optionally restricted to certain days of the week.

You could use a trigger like this to set up your Mac environment before arriving at work, run periodic maintenance or backup scripts late on the weekend, or launch iChat for your weekly video conference.

Keep in mind that the Mac itself must be awake for the macro to run – you can configure a time to wake from sleep in System Preferences. Also, if the screen is screen saving or the display is asleep, UI actions like typing keystrokes or selecting menus will not work. The actions that wake the display and/or stop the screen saver may be useful to resolve some of these issues.

While Logged In

You can use the While Logged In trigger to repeatedly execute a macro during a portion of the day, optionally restricted to certain days of the week.

You could use a trigger like this to run periodic maintenance or track changes.

Macro Palette

You can have a macro execute when you click on it in a floating Macro Palette. Keyboard Maestro will only display the palette when there are active Macros, so if your Macros are restricted to particular applications, then the Macro Palette will only appear in those applications. This is particularly useful for less frequently used macros whose Hot Key you might forget.

You can control the sorting order of macros by adding two characters and a closing parenthesis (eg “01)My

Macro”). The prefix will be removed before displaying in the macro palette, but will be used to control the order of the macros shown.

You can edit a macro by holding down the option key and selecting it from the macro palette.

Status Menu

You can have a macro execute when you select it from the Keyboard Maestro Status Menu (accessed by the Keyboard Maestro icon on the right hand side of the menu bar). You add a macro to the Status Menu by including the Status Menu trigger as a Macro trigger.

Keyboard Maestro will only display the Status Menu triggered macros that are active, so if your Macro is restricted to particular applications, then it will only appear in in the Status Menu while those applications are active.

The Status Menu is particularly useful for less frequently used macros whose Hot Key you might forget.

You could use a trigger like this to add custom facilities to applications, such as to open specific common files or set up windows in specific ways.

You can control the sorting order of macro groups and macros by adding two characters and a closing bracket (eg “01)My Macro”). The prefix will be removed before displaying in the status menu, but will be used to control the order of the macros shown.

You can edit a macro by holding down the option key and selecting it from the status menu.

Public Web

Keyboard Maestro has an built-in web server. You can enable it in the Web Server preference pane. If enabled, and if you configure a username and password, you can connect to your Keyboard Maestro's web server and login and then execute any macro you have defined. Also, if the web server is enabled, and if you have configured any Macro with a Public Web trigger, then anyone on the Internet can connect to your Mac and trigger Public Web macros.

Macros are only available if they are currently active (ie, they must not be disabled or in a Macro Group that is disabled, and their Macro Group must be currently active at the time).

For example, if you are running some sort of process on your Mac that occasionally fails, you could write a script to restart it and make it available as a Public Web triggered Macro, which you (or anyone else) could then execute to restart the process.

Clearly there are some serious security issues with this, so you should use a lot of caution when you allow any macro to be executed with a Public Web trigger.

Mounted Volume

Keyboard Maestro can trigger a macro when a volume (disk) is mounted or unmounted.

You could use this trigger to perform actions when you connect a backup disk, or automatically sync files with a disk when it is mounted.

USB Device

Keyboard Maestro can trigger a macro when a USB Device is attached or detached.

A great use for this trigger is to launch your scanner software when you turn on your scanner, and quit it when you turn off your scanner.

Wireless Network

Keyboard Maestro can trigger a macro when you connect or disconnect from a specific wireless network.

Since you typically have a well known specific wireless network at home and at work, you can use this trigger with a MacBook to configure your Mac for work or home use. An obvious use for this is to change your network location when you connect to a wireless network using the Set Network Location action.

Keep in mind that you will disconnect and reconnect your wireless network every time you sleep and wake your Mac. You can get the current Wireless Network Name(s) or Network Location using the appropriate

[Text Tokens](#).

Device Key

This is an advanced trigger—generally you should use a [Hot Key Trigger](#) if possible.

Keyboard Maestro can trigger a macro when any device key is pressed—this includes modifier keys, mouse buttons, programmable keyboards like [P.I. Engineering's X-Key](#), and even the brightness buttons on USB connected monitors.

The macro can be executed when the key is pressed, released or repeat continuously while it is held down. The trigger can also optionally be restricted to when certain modifier keys are pressed.

For example, a macro could fire every five seconds while Control-Mouse Button 3 is held down.

Unlike hot key triggers, the pressed key is not removed or affected in any way. This trigger watches input devices at a low level, but it does not affect them, so any key presses continue to have their normal operation as well as triggering the macro. This is fine for modifiers, unused mouse buttons, programmable keyboards and other unused buttons, but would likely be problematic for normal keystrokes which will continue to have some other, probably unwanted, affect.

MIDI Note

Keyboard Maestro can trigger a macro when it receives a MIDI note. You execute a Macro by pressing a key on a MIDI device like an electronic Keyboard.

The macro can execute when the MIDI note is pressed (note on), released (note off) or continuously while it is held down. This allows you to do things like have a macro execute when the key is pressed, and then a second macro execute when the key is released, for example to toggle a setting on and then off again.

By Script

Keyboard Maestro can be triggered from an AppleScript or shell script.

```
tell application "Keyboard Maestro Engine"
  do script "[Name or UID of Your Macro]"
end tell
```

or

```
osascript -e 'tell app "Keyboard Maestro Engine" to do script "[Name or UID of Your Macro]"'
```

Select from the Or by Script menu to show example code for triggering a macro from Apple Script, shell script, Perl, Python or Ruby.

The macro must be defined and currently active, meaning it must be enabled and the Macro Group that contains it must be enabled and active.

Macro Actions


- [Overview](#)
- [Application Control](#)
- [Clipboard Actions](#)
- [Control Flow Actions](#)
- [Debugger Actions](#)
- [Mail Control Actions](#)
- [Execute Actions](#)
- [File Actions](#)
- [Safari and Google Chrome Actions](#)
- [Image Actions](#)
- [Interface Control](#)
- [iTunes Control](#)
- [Keyboard Maestro Actions](#)
- [MIDI Actions](#)
- [Notification Actions](#)
- [Open Actions](#)
- [QuickTime Player Control](#)
- [Switcher Actions](#)
- [System Control](#)
- [Text Actions](#)
- [Variable Actions](#)
- [Prompt For User Input Action](#)

- [Web Actions](#)

Overview

A Macro executes a sequence of [Macro Actions](#) in order. There are many actions to choose from, and these are detailed below. Some simple actions, such as Sleep Computer, require no other information and simply do their job, while other more complex actions, such as Select Menu Item, require you to specify more information, such as a target application or menu name.

There are many powerful [Macro Actions](#) available for your use, and you can sequence them together to perform complex tasks.

To add an action, edit your macro, click the New Action button, or equivalently the  button below the detail view. This will show the lists of possible actions.

To see all actions, select the [All Actions](#) category. To select just your favorite actions, select the [Favorites](#) category. You can drag actions into your Favorites category. To see plug in actions that you have added to Keyboard Maestro, select the [Third Party Plug Ins](#) category.

Double click or drag one or more of the actions to add them to the action list for the currently edited macro.

Some actions are simple and do not require any configurion (eg Shut Down Computer). Other actions require you to specify parameters to the action, such as which window to move and by how much. You do this by editing the values for the action.

In Keyboard Maestro, there are two kinds of processed fields in actions, text token fields and numeric calculation fields. Most text fields are text token fields. Most numeric fields are calculation fields.

Text token fields typically (but not always) have an Insert Token pull down menu, and are always somewhat long. They are designed to allow you to enter a defined text item, rather than a numeric value. In these fields you use text tokens, which are marked with percents, like %CurrentClipboard% or %Variable%My Variable%. You can insert a text token from the Insert Token popup menu, or by choosing [Insert Token](#) from the [Edit menu](#), see the [Text Tokens](#) section.

Numeric calculation fields are usually small to start off with (assuming they contain only a number), and usually have a stepper (double arrows) next to them. They are designed to allow you to enter a specific numeric value. In these fields, you can type a raw calculation. As soon as you type anything other than a digit, they expand to a larger size to allow for a calculation. Calculation fields never use percent tokens, but they can include variables or functions. You can insert a function or variable by choosing [Insert Function](#) from the [Edit menu](#), see the [Calculations](#) section.

If desired, you can include a calculation in a text token field by using a %Calculate% token like [%Calculate%1+2%](#). You can never use text tokens in a calculation field.

Application Control

Application Control actions allow your to switch, quit, or hide applications. The actions are:

Activate Last Application

switches back to the previous application you were in.

Activate Next Application

switches to the next application (alphabetically).

Activate a Specific Application

launches if necessary and brings the specified application to the front, optionally reopening the initial windows. If the application is already at the front, the action can optionally switch out, hide or quit it, allowing you to toggle an application. Very useful for utilities like [PCalc](#).

Bring Application Windows to Front

brings all the windows of the current application to the front.

Quit All Applications

quits all foreground applications, optionally honoring the excluded application list.

Quit Other Applications

quits all foreground applications except the current one, optionally honoring the excluded application list.

Quit a Specific Application

quits the specified application, optionally force quitting or relaunching.

Hide All Applications

hides all foreground applications, optionally honoring the excluded application list.

Hide Other Applications

hides all foreground applications except the current one, optionally honoring the excluded application list.

Hide Front Application

hides the current application.

Hide a Specific Application

hides the specified application.

Show All Applications

shows all foreground applications.

Show a Specific Application

shows the specified application.

Clipboard Actions

Clipboard actions let you manipulate the system clipboard, Named Clipboards, the clipboard history and cut, copy or paste clipboard items. The actions are:

Cut, Copy, Paste

simulate Command-X, Command-C, or Command-V keystrokes to Cut, Copy or Paste to/from the system clipboard.

Set Clipboard to Past Clipboard

set the system clipboard to a previously copied item.

Set Clipboard to Text

set the system clipboard to plain or styled text. Text Tokens are provided to include such things as the time or date.

Set Clipboard to Variable

set the system clipboard to the value of a variable.

Delete Current Clipboard

delete the current clipboard, replacing it with the previously copied value from the clipboard history. This can be used to restore the system clipboard after another action sets the clipboard temporarily.

Delete Past Clipboard

delete a past system clipboard. This can be used to restore the system clipboard after another action sets the clipboard temporarily.

Cut, Copy, Paste to/from a Named Clipboard

cut, copy or paste to/from the specified Named Clipboard.

Copy (Named) Clipboard to (Named) Clipboard

copy the system or a Named Clipboard to the system or another Named Clipboard.

Apply Style to (Named) Clipboard

apply a style (like font or size or underline) to (a section of) the system or a Named Clipboard.

Apply a BBEdit Text Factory

apply any saved BBEdit Text Factory to the system or a Named Clipboard.

Filter (Named) Clipboard

apply one of a set of Filters to the system or a Named Clipboard.

Search and Replace (Named) Clipboard

search and replace the system or a Named Clipboard, optionally using regular expressions, and allowing Text Tokens. The replacement can also include \$1 tokens for regular expression replacements.

Search (Named) Clipboard

search the system or a Named Clipboard using regular expressions, and capturing the results.

Substring of (Named) Clipboard

extract a substring of the system or a Named Clipboard.

Display (Named) Clipboard

display the system or a Named Clipboard in a window.

Control Flow Actions

You can control the flow of a macro execution using a variety of actions, from a simple Pause for a number of seconds, through a complex nesting of If/Then/Else and looping.

The actions to control the flow of a macro are:

Pause

pause for a number (which may be a calculation) of seconds.

Pause Until

pause until conditions are met.

Until

execute a list of actions until conditions are met.

While

while conditions are met, execute a list of actions.

Repeat

repeat a list of actions a number (which may be a [calculation](#)) of times.

If Then Else

if conditions are met, execute a list of actions, otherwise execute another list.

For Each

loop over a collection of values.

Execute a Macro

execute another macro (like a subroutine).

Cancel All Macros

Cancel all macros that Keyboard Maestro Engine is currently executing.

Cancel This Macro

Cancel this macro (including any macro that executed this macro).

Cancel Just This Macro

Cancel just this macro, continue executing a macro that executed this macro.

Break From Loop

Cancel execution of the current loop, and continue execution after the loop.

Semaphore Lock

Wait and lock a semaphore to ensure exclusive access/execution.

Semaphore Unlock

Unlock a semaphore (happens automatically if macro terminates).

Semaphore Reset

Forcefully reset a semaphore.

You will often need to use the Pause action to slow down replaying of an action sequence to allow the system time to catch up—especially after you have changed applications or if you want to use a mouse click.

The condition clause of the flow control actions can be any of:

- Any of the following are true – at least one condition must be true.
- All of the following are true – every condition must be true.
- None of the following are true – no condition is true.
- Not all of the following are true – at least one condition must be false.

If there are no conditions in the set at all, the action will not execute anything except the Until action which will execute the actions once. Neither side of the If Then Else will execute.

There is over a dozen different kinds of conditions, from testing what the current application is or examining variables or the clipboard, to checking your network location or testing a pixel on your screen, see the [Conditions](#) section.

The For Each action loops over a set of collection items, setting a variable to each value and executing the contained actions. The collections can include:

- A number range (eg 1 to 10).
- The running applications.
- The files in a directory (optionally recursively).
- The files or folders selected in the Finder.
- The currently mounted volumes.
- The lines in a clipboard, Named Clipboard, variable or file.
- The substrings in a clipboard, Named Clipboard, variable or file.
- The clipboard history.

The control flow actions include a nested list of their own actions to execute, and that nested list can include further control flow actions—go wild! But keep in mind, there may be a time when a shell script or AppleScript is a more useful way of describing your solution.

See also the [Conditions](#), [Variables](#) and [Calculations](#) sections.

Debugger Actions

Keyboard Maestro includes a built-in [Macro Debugger](#) which allows you to observe and control the progress of a macro in action. As well as using the Macro Debugger normally, you can use any of these actions, triggered by any method you like, to control the debugger.

Start, Finish, or Toggle

start or finish debugger, or toggle the debugger on or off.

New Macros Paused or Run

control whether new macros start paused (while debugging).

Breakpoint This, All or Other Macros

pause the specified macros in the debugger.

Step Over, Into or Out Other Macros

step other currently executing macros.

Continue This or Other Macros

resume executing specified macros.

Note that regardless of whether a macro would otherwise be paused, these actions will always execute. So, for example, even if you have New Macros Paused, macros that start with these debugger actions will still execute the action.

See also the [Macro Debugger](#) section.

Mail Control Actions

Mail Control actions allow you to interact with Apple Mail, send messages or set mail flags. The actions are:

Send Mail Message

send (or create) an email message, complete with optional attachment.

Set Mail Flag/Flagged/Read/Junk Status

set the flags on the currently selected mail message.

Note that the currently selected mail message is not necessarily the front window, it may be a message selected in the viewer behind the front window, especially if the front window is a reply or new message.

Execute Actions

Execute actions let you execute AppleScripts, shell scripts, Automator Workflows, JavaScripts or another macro. Shell scripts can be any kind of script: sh, zsh, tcsh, perl, python, ruby, and so on.

The actions are:

Execute AppleScript

execute a specified AppleScript, either from a file or text. For example, `say "hello"`.

Execute a Shell Script

execute a specified shell script, either from a file or text. For example, `pbpaste | pbcopy`.

Execute JavaScript in Safari or Google Chrome

execute a specified JavaScript, either from a file or text. For example, `alert("hello")`.

Execute Automator Workflow

execute a specified Automator Workflow.

Execute a Macro

execute a specified macro. This allows you to create subroutines of instructions. The current macro waits until the submacro finishes, or optionally execute asynchronously while the macro continues on.

The results of an AppleScript, shell script or JavaScript can be:

- Ignored.
- Displayed in a floating window.
- Displayed briefly in a [Notification Center](#).
- Typed in to the current selection.
- Pasted in to the current selection.
- Saved in to a variable.
- Saved in to the system or a [Named Clipboard](#).
- Asynchronously ignored - the action runs while the macro continues on.

For example, you could have a shell script `date` display briefly in the [Notification Center](#) every hour, or use a hot key to type the results directly into your text editor.

These powerful actions allow you to add any new facilities we have not provided for, stringing them together with other actions as desired.

You can also use the clipboard by piping from `pbpaste` and to `pbcopy`.

Shell scripts are executed in the background and can access variables by using environment variables, see the [Variables](#) section.

AppleScripts are executed in the background via `osascript`. This means they are not allowed to do user interaction. You can work around this by asking an application like System Events to do the user interaction for you, for example:

```
tell application "System Events"
  activate
  display dialog "Hello"
end tell
```

AppleScripts can access variables by using environment variables (using `system attribute`) or by talking to the Keyboard Maestro Engine, see the [Variables](#) section. Note that AppleScript's `system attribute` is not safe for international characters, although can use code like:

```
set v to do shell script "echo $KMVAR_Variable"
```

JavaScript can access variables by using the document.kmvar dictionary, see the [Variables](#) section.

It is a short step from executing a script action to writing some [Plug In Actions](#).

File Actions

File actions allow you to interact with the file system, moving, copying, duplicating, trashing or deleting files or folders. The actions are:

- Reveal a file.
- Create a new folder.
- Move (or rename) a file or folder.
- Copy a file or folder.
- Duplicate a file or folder.
- Trash a file or folder.
- Delete a file.
- Delete a file or directory (recursively). This is potentially very dangerous.
- Read a file to a variable or to the system or a [Named Clipboard](#) (can be an image or styled text).
- Write to a file from a variable or the system or a [Named Clipboard](#) (can be an image or styled text).
- Append text to a file from a variable or the system or a [Named Clipboard](#).
- Create a new folder.
- Get or Set file attributes.

The source path must be an absolute path (or a home relative `~` path). The destination path must either be a simple single path component (not `~` or `/`) or an absolute path. If it is a simple path component, then it is relative to the parent directory of the source path. This is particularly useful in the Move or Rename action, as you can, for example, rename from `~/Folder/Old Name` to `New Name`.

You can read or write files in a variety of formats, including PNG, TIFF, JPEG, as well as HTML, Web Archive, Word Document and more.

File attributes include:

- file type (*r/o*).
- file size (*r/o*).
- creation and modification date.
- owner name and ID (*r/o*).
- group name and ID.
- posix permissions.
- whether the extension is hidden.
- HFS creator and type codes.
- Mavericks tags (set, add, remove or toggle).
- parent path, file name, base name, and extension (*r/o*).

Safari and Google Chrome Actions

You can control Safari or Google Chrome, including creating windows and tabs, moving tabs, and interacting with web pages. The actions are:

- New window with optional URL.
- New tab with optional URL.
- Next or previous tab.
- Select a specific tab.
- Wait for the current tab to finish loading.
- Set the URL or title.
- Click a link.
- Focus or select a field.
- Set a variable to the contents of a field.

- Set a field, checkbox or radio button.
- Submit or reset a form.
- Execute arbitrary JavaScript and return the results.

There are a number of [Text Tokens](#) to get various values from the web browser, including title, URL, ready state, a field value or the result of arbitrary JavaScript.

There are also a number of [functions](#) to get various values from the web browser, including the tab count, tab index, and whether the browser has currently finished loading.

Note that complex web sites often report that they have finished loading and then start loading again, which is why the action to wait for the browser to finish allows you to specify a required minimum time.

See also the [Text Tokens](#) and [Calculations](#) sections.

Image Actions

Image actions allow manipulations such as flipping or rotating images, or capturing the image of a window or screen. The actions are:

- Screen Capture a window, screen or all screens.
- Find Image on Screen optionally highlighting it.
- Read an image from a file.
- Write an image to a file.
- Create New Image with size and color.
- Flip Image horizontally or vertically.
- Rotate Image by 90°, 180°, 270° or an arbitrary angle.
- Resize Image, resize canvas, add or remove margins or crop.
- Composite images or styled text onto an Image.
- Draw a Shape (line, rectangle or oval) onto an Image.
- Trim transparent areas from the edge of the image.
- Display an image in a window.
- Get Image Size into a variable.

Image actions typically act on an image in the system or a [Named Clipboard](#). You can read or write images files using the File actions.

Interface Control

Interface Control actions allow you to interact with the user interface, selecting menus, clicking buttons, simulating keystrokes and so forth. The actions are:

Manipulate Window

resize, move, center, bring to front, close, zoom, or minimize a specified window in a specified application.

Bring Application Windows to Front

brings all the windows of the current application to the front.

Move or Click Mouse

move, click or move and click the mouse using any mouse button, and optionally dragging. You can specify modifiers such as the shift key, as well as the position, relative to any corner of the front window, the main screen, or the current mouse location, or a found image on the screen.

Select or Show Menu Item

select or show a specified menu item in the front or a specified application.

Press Button

press a named button in the front window.

Simulate Keystroke

simulate pressing a specified keystroke.

Simulate Scroll Wheel

simulate scrolling the mouse wheel up or down, left or right.

Use Variable

use a variable to move the mouse, or to adjust a window or the front application, or the system volume.

The various size and location fields can be [Calculations](#).

The Manipulate Window action includes defaults for positioning a window in various columns or corners of a screen. This is a useful way to see some examples of calculations you can use.

The Move or Click Mouse has lots of options, including single, double or triple clicking, and where the click is relative to. If you record a mouse click, you can (for a short while) adjust the corner that the click is relative to and the values will adjust accordingly. You can also perform a click and drag with this action.

The Select Menu Item action lets you choose from all currently running applications and their menus, which

helps ensure you have correctly specified the menu. If you leave the menu item blank, Keyboard Maestro will show you the menu and let you then select it manually.

The Select Menu and Press Button actions allow you to specify multiple options separated by a vertical bar (eg Show/Hide) to allow for toggling menus. They will also ignore the difference between three dots (...) and an ellipsis so you do not have to worry which one the menu uses. Alternatively, you can start the name with an ^ and use a regular expression to match the menu or button name. Also, Select Menu will translate the word "APPLICATION" (all capitals) into the current application name, allowing menu selections like APPLICATION -> About APPLICATION.

The Select Menu and Press Button actions will, by default, abort the macro if the action is not successful (for example if the button or menu cannot be found or is disabled). The action can be configured to allow the macro to continue if the menu or button is not essential (such as “Mark As Read” which might be disabled if the item is already marked as read).

iTunes Control

iTunes Control actions allow you to interact with iTunes, playing songs, stopping or pausing, rewinding or fast-forwarding. All actions will launch iTunes if it is not already running. The actions are:

Play a Specific Track

play a specified song.

Play a Specific Playlist

play the songs in a specified Playlist.

Play a Random Track

play a random song.

Play a Random Track from a Specified Playlist

play a random song from a specified Playlist.

Play/Pause Current Track

toggle from playing to pausing or vice-versa.

Pause Current Track

pause the current song.

Stop Current Track

stop playing any song.

Fast-Forward Current Track

fast-forward the current song. It will keep fast-forwarding until you do something else or until it reaches the end of the song.

Rewind Current Track

rewind the current song. It will keep rewinding until you do something else or until it reaches the beginning of the song.

Next Track

play the next song.

Previous Track

Go to the previous song or the start of the current song if it is already playing.

Increase/Decrease or Set iTunes Volume

Increase, decrease or set the iTunes volume.

Increase/Decrease or Set Rating

Increase, decrease or set the rating of the current track.

Keyboard Maestro Actions

These actions allow you to control how Keyboard Maestro behaves. The actions are:

Record Quick Macro

record a temporary macro for immediate playback.

Trigger Macro by Name

type the name of a macro to trigger.

Set Macro or Group Enable

enable, disable or toggle a Macro or Macro Group.

Activate a Macro Group

activate a Macro Group for one action or toggle activation.

Show Macro Group

show a Macro Group as a palette for one action or toggle visibility.

Cancel Macros

cancel one, other, or all macros.

Comment

does nothing, just allows you to comment a macro.

Show, Hide or Toggle the Global Macro Palette

shows or hides the global macro palette.

Show Status Menu

Shows the Keyboard Maestro status menu.

Trigger Macro by Name allows you to trigger any active macro by name. By default, Keyboard Maestro creates a macro triggered by Command-Control-Option-T or the status menu which triggers a macro by name.

Recording allows Keyboard Maestro to watch you as you perform a task and create the actions to produce a similar result.

Record Quick Macro is a variant of this that you can activate at any time without launching Keyboard Maestro. You trigger the recording, and then perform a series of actions, and then turn off recording. Then press the associated hot key to replay the recording.

By default, Keyboard Maestro creates a macro triggered by Control-F1 which executes the Record Quick Macro action with a hot key of Option-F1. So for example, if you press Control-F1 to start quick recording, type “hello” and then press Control-F1 again to finish recording, then Keyboard Maestro will type “hello” each time you press Option-F1.

Because you cannot see or edit the recorded actions, it is best to keep them very simple. A good rule of thumb would be not to touch the mouse, just use the keyboard.

For example, say you wanted to quote a dozen different words in a paragraph, you could click in the middle of the first word, press Control-F1 to start quick recording, type Option-Left Arrow, quote (“), Option-Right Arrow, quote (“), and then Control-F1 again to finish recording. Now click in the middle of each remaining word and press Option-F1.

The Set Macro or Group Enable action allows you to enable, disable or toggle the enable of a macro group or macro. The Keyboard Maestro editor does not need to be running, but it will see the enable state when it is next launched (or immediately if it is already running).

If you hide the global macro palette, it will remain hidden until shown. You can also hide the global macro palette by clicking the close icon, so you may wish to include a global macro, perhaps with a status bar trigger, which uses the Show Macro Palette to redisplay it.

MIDI Actions

Send a MIDI message:

Send MIDI Note On

sends a MIDI Note On message, specifying the note, velocity and channel.

Send MIDI Note Off

sends a MIDI Note Off message, specifying the note, velocity (usually 0) and channel.

Send MIDI Control Change

sends a MIDI Control Change message, specifying the control, value and channel.

The MIDI messages come from a device named “Keyboard Maestro” which will be created as soon as you add any MIDI action.

Notification Actions

Keyboard Maestro can notify you in a variety of ways:

- Notification: displays a message in the Notification Center.

Growl

display a message via Growl (or in a popup HUD window if Growl is not installed).

Display Text

display text (which may contain Text Tokens).

Alert

display an alert (which may contain Text Tokens).

Prompt For User Input

display a dialog requesting information, see the Prompt For User Input Action section.

System Beep

play a standard system beep.

Play Sound

play a sound, optionally through a specific output device.

Speak Text

speak text (which may contain Text Tokens) in any desired voice or speed.

Log

log a message to the Engine.log file.

Highlight Location

highlight a location on the screen.

Open Actions

Open actions allow you to open files, folders, URLs or System Preference Panes. The actions are:

Open File or Folder

opens a specified file or folder. Don't underestimate the power of this command as you can open bookmarks or other action documents to perform a lot of customized actions.

Open the Finder Selection

opens the files or folders that are currently selected in the Finder.

Open URL

open a URL with the appropriate helper.

Open System Preference Pane

opens a specified System Preference Pane.

Files, folders, the Finder selection and URLs can all open either in their default applications, or a specific application. So you could, for example, create a macro that opens the Finder selection in BBEdit.

QuickTime Player Control

QuickTime Player Control actions allow you to interact with QuickTime Player, playing movies, stopping or pausing, stepping forward or backward, or adjusting the volume. All actions will launch QuickTime Player if it is not already running. The actions are:

Play Current Movie

play the current movie.

Play/Pause Current Movie

toggle from playing to pausing or vice-versa.

Pause Current Movie

pause the current movie.

Step Forward Current Movie

step the current movie forward one frame.

Step Backward Current Movie

step the current movie backward one frame.

Increase/Decrease or Set QuickTime Player Volume

Increase, decrease or set the QuickTime Player volume.

Switcher Actions

Keyboard Maestro includes several powerful switchers, including Application Launcher, Application Switcher, Window Switcher, and Copy, Cut and Paste Clipboard Switchers and Clipboard History Switcher.

Each of these switchers is actually just a macro action, triggered like any other macro. By default Keyboard Maestro creates a Macro Group called Switcher Group which includes the following macros:

Activate Application Launcher

activates the Application Launcher (Command-Control-Tab).

Activate Application Switcher

activates the Application Switcher (Command-Tab).

Activate Clipboard Copy Switcher

copies the current selection to a named clipboard you select (Command-Shift-C).

Activate Clipboard Cut Switcher

cuts the current selection to a named clipboard you select (Command-Shift-X).

Activate Clipboard Paste Switcher

pastes a named clipboard you select into the current system clipboard and current selection (Command-Shift-V).

Activate Clipboard History Switcher

pastes a previous system clipboard you select from the clipboard history into the current system clipboard and current selection (Command-Control-Shift-V).

Activate Window Switcher

activates the Window Switcher (Control-Tab).

You can create your own macros using these actions or adjust their triggers within the Switcher Group.

By default, Keyboard Maestro overrides Command-Tab and the system application switcher. If you wish to use the default system switcher with Command-Tab, you can do so by disabling Keyboard Maestro's macro or by changing its hot key trigger.

System Control

System Control actions allow you to control your Mac. Most are pretty self-explanatory. The actions are:

Put Computer to Sleep

puts your Mac to sleep.

Restart Computer

restarts your Mac.

Shut Down Computer

shuts down your Mac.

Fast User Switcher

Fast User Switch to a specific user designated in the macro.

Log Out

Log Out the current user.

Set Find Pasteboard

sets the global "Find" pasteboard, typically equivalent to selecting text and choosing Use Selection To Find. The Find Pasteboard is used in most applications.

Open/Close CD Tray

Opens the CD tray if it is closed, or closes it if open (especially useful for keyboards that do not have an eject key).

Toggle System Sound Mute

Mutes the sound if sound is currently on, unmutes the sound is already muted.

Increase/Decrease System Sound Volume

Increases or decreases the sound volume.

Set System Sound Volume

Sets the sound volume to a specific amount (can be a [calculation](#)).

Increase/Decrease Screen Brightness

Increases or decreases the screen brightness.

Start or Stop Screen Saver

Starts or stops the system screen saver.

Sleep or Wake Screen

sleeps or wakes the monitor.

Set Network Location

sets the network location.

Some of these actions require particular hardware support and so may not work on all Macs.

Text Actions

Most of these actions are covered in their respective sections (eg Set Clipboard to Text is covered in the [Clipboard Actions](#) section).

The Insert Text action deserves special attention as it allows you to insert specified text by

- typing the plain text as a sequence of keystrokes.
- setting the system clipboard to plain text and pasting.
- setting the system clipboard to styled text and pasting.

For inserting text by typing, the text may include non-ASCII characters as long as they can be typed on the keyboard with one or two keystrokes.

The Display Text action allows you to display the resulting text in a floating window or [Notification Center](#).

Tokens are provided to include variables, calculations, the time or date, and a large variety of other system information.

The Insert Text action can be very useful to insert standard text templates, such as your signature, address, copyright or other boilerplate text, and so on. If you ever get email from Peter, you will probably notice that many of his emails end with "Thanks for your support, Peter." – with the amount of email he deals with, you don't think he types that in every time, do you?

Insert Text can expand various tokens, including dates in any [ICU date format](#). For example you can use an

Insert Text action to insert the copyright message at the top of code files:

```
/*
 * Created by %UserName% on %ICUDateTime%EEE d MMM yyyy%.
 * Copyright (c) %ICUDateTime%yyyy% Stairways Software. All rights reserved.
 */
```

There are three ways to type text in Keyboard Maestro: Insert Text by Typing, Insert Text by Pasting, and Type Keystroke. A common question is when should you use each?

The Type Keystroke action lets you type a single keystroke, pretty much any kind of keystroke you can type on the keyboard.

The Insert Text by Typing action converts plain (simple) text into keystrokes and then types each of them individually. It only works with characters that can be typed on the keyboard with one or two keystrokes (eg Option-e e usually types é, and Keyboard Maestro can duplicate that typing).

The Insert Text by Pasting action sets the clipboard to the (optionally styled) text and then types a Command-V keystroke.

Each action has its advantages and disadvantages:

The Type Keystroke action is just like typing yourself, so you can type any keystroke and any number of them, but it can be tedious to create a sequence of keystrokes in a macro.

The Insert Text by Typing action is also just like typing yourself, similarly permitting characters like tab/return and maintaining the current style in the program into which you are typing, but it is slow for large amounts of text and can't type exotic characters like Emoji.

The Insert Text by Pasting action is fast for large text and can include any kind of characters, as well as styles like bold and italic, but it overwrites your clipboard and its handling of styles may occasionally not conform to your expectation.

So you should use Type Keystroke when:

- You are entering a single keystroke and perhaps need particular control over the exact keystroke typed, or
- The keystroke includes Command or Control (and sometimes Option) keystrokes.

Use Insert Text by Typing whenever:

- The text consists of plain text characters that can be typed on the keyboard, and
- The text is relatively short (up to say 30 or 40 characters), or
- You want to use characters like Return or Tab to perform actions like moving to the next field.

Use Insert Text by Pasting when:

- You don't mind the clipboard being overwritten, and
- The text is long, or
- The text contains exotic untypable characters, or
- The text contains returns or tabs which you don't want to perform actions like moving to the next field.

See also the [Text Tokens](#) section.

Variable Actions

Keyboard Maestro includes permanently stored variables that you can use in a number of special purpose actions, as well as in [Calculations](#) or [Text Tokens](#) which can be used in almost any field in Keyboard Maestro. These specific actions let you manipulate variables:

Set Variable to Text

set a variable to text. [Text Tokens](#) are provided to include such things as the value of other variables or the time or date.

Set Variable to Calculation

set a variable to the result of a calculation with numeric formatting, see the [Calculations](#) section.

Set Variable to Keychain Password

set a variable to a password value from the Keychain.

Set Keychain Password to Text or Variable

the reverse, sets a password in the keychain to a value.

Filter Variable

apply one of a set of filters to a variable, see the [Filters](#) section.

Search and Replace Variable

search and replace the value of a variable, optionally using [regular expressions](#), and allowing [Text Tokens](#). The replacement can also include \$1 tokens for [regular expression](#) replacements.

Search Variable

search a variable using [regular expressions](#), and capturing the results.

Substring of Variable

extract a substring of a variable.

Use Variable

Use a variable to set a variety of system values like mouse location or front window. See below.

Prompt For User Input

display a dialog asking for a variety of user input, see the [Prompt For User Input Action](#) section.

The Use Variable action can perform the following actions:

- Set the mouse location.
- Set the front window position, size or frame.
- Set the front window by name.
- Set the front application by name.
- Set the system volume.

Most of these have analogs in the [Text Tokens](#), so for example you might do:

```
Set variable 'Temp' To Text '%FrontWindowFrame%'
Set variable 'Temp' To Calculation 'Temp[1]+Temp[3]/4,Temp[2]+Temp[4]/4'
Use 'Temp' to Set the Mouse Location
```

Which will result in the mouse being placed in the middle of the upper left corner of the front window.

Search Variable will use a [regular expression](#) to pull apart a variable, allowing you to extract the various parts. It displays the results live in the editor, allowing you to construct complex [regular expressions](#) more easily.

See also the [Variables](#), [Text Tokens](#) and [Calculations](#) sections.

Prompt For User Input Action

Prompt For User Input will allow you to ask for user input, storing any results in variables as designed. If the variable is a password variable (ie, its name starts or ends with "Password" or "PW"), it will be displayed in a password field.

If the default value you specify consists of choices separated by a bar (|), then a popup menu will be used (the first value will be the default value, and can be repeated later if a different location is desired). For example "Better|Good|Better|Best" would result in a popup menu with Good, Better, and Best, with Better pre-selected.

If a component of a popup menu item is a dash (-), then a separator is displayed. For example, "Good|Better|Best|-|Terrible"

If a component of a popup menu item starts with a prefix followed by two underscores, then the prefix will be used as the variable value, while the latter part will be displayed in the menu. For example, "B__Good|F__FaillC__OK|B__Good|A__Excellent".

If the field starts with a bar (|), then token expansion happens first, followed by separating by bar (|) - this allows you to have a variable list of entries, for example, "IThis|%Variable%Multiple Entries%". Otherwise, the entry is separated by bar first, and then token expansion is applied to each field, so that fields may contain bars, for example, "This|That|%Variable%One Entry%".

If the popup menu would contain only "0" and "1", then a checkbox is used (so for example, "0|1" or "1|0", depending on the desired default).

If the variable name starts with a prefix followed by two underscores, then the label will show only the latter part of the variable name (for example, `[MyMacro__Text Message]` will display a label of `[Text Message]`).

You can include one or more buttons, which may optionally cancel the macro. You can also include /«letter» to configure command keys for the buttons (/ means escape, / by itself means the default button) (for example, "Help/H". If there are no text fields in the dialog, the command key is not needed and the letter by itself will complete the dialog. The name of the button pressed will be stored in the `[Result Button]` variable.

Prompt for User Input 'Example'

Try

Title: Example

Prompt: Please enter the details for these variables.

Insert Token ▾

Variables and Default Values:

| | | | | |
|--|--|----------------------|---|----------------------------------|
| | | Normal Text | ▾ | default value |
| | | Password Variable | ▾ | default password |
| | | Popup Menu Default | ▾ | Better Good Better Best |
| | | Popup Menu Separator | ▾ | Good Better Best - Terrible |
| | | Popup Menu Values | ▾ | F_Fail C_OK B_Good A_Excellent |
| | | With Variable | ▾ | This That %Variable%One Entry% |
| | | Variable List | ▾ | This %Variable%Multiple Entries% |
| | | Check Box On | ▾ | 1 0 |
| | | Check Box Off | ▾ | 0 1 |
| | | MM__This Variable | ▾ | Note MM is not shown |

Buttons:

| | |
|---------------------------------------|--|
| <input type="button" value="OK"/> | <input type="checkbox"/> Cancel Macro |
| <input type="button" value="Cancel"/> | <input checked="" type="checkbox"/> Cancel Macro |
| <input type="button" value="Help/H"/> | <input type="checkbox"/> Cancel Macro |

Example

Please enter the details for these variables.

Normal Text: default value

Password Variable:

Popup Menu Default: Better ▾

Popup Menu Separator: Good ▾

Popup Menu Values: Fail ▾

With Variable: This ▾

Variable List: This ▾

☒ Check Box On

☐ Check Box Off

This Variable: Note MM is not shown

| |
|----------|
| Good |
| ✓ Better |
| Best |

| |
|----------|
| ✓ Good |
| Better |
| Best |
| Terrible |

| |
|-----------|
| ✓ Fail |
| OK |
| Good |
| Excellent |

| |
|--------------|
| ✓ This |
| That |
| Test Entry |

| |
|--------|
| ✓ This |
| Good |
| Better |
| Best |

See also the [Variables](#) section.

Web Actions

You can open a URL in your default browser or a specified alternative and you can have Keyboard Maestro query you for a search term and then search a specified web site (by default Google).

Macro Syncing

Keyboard Maestro is licensed on a per user basis on up to five Macs, so if you use it on two or more Macs you may want to transfer some of your macros from one to the other. You can do this by selecting the desired macros and choosing [Export Macros](#) from the [File menu](#) to export your macros, and then importing them on the target Mac.

Alternatively, you may want (almost) all your macros on both Macs, in which case you can set up your macros to sync between them. This means any change you make on one Mac will be mirrored on the other

and vice versa, although you generally should not edit your macros simultaneously on both Macs.

To do this, you need a file location that is mirrored on both Macs. DropBox or a similar service is good for this, or you can use a shared file server, although you must ensure it is available at all times to both Macs.

You are then ready to start syncing your macros.

WARNING: Syncing macros is an all or nothing affair, so any macros on the target Mac before you start syncing will be lost.

On the Mac that currently has your macros, choose [Start Syncing Macros](#) from the [File menu](#). Read the text carefully, and then click the [Create New](#) button. Save your existing macros in the macro sync file in your shared location. From now on, Keyboard Maestro will sync any changes to/from that file.

Wait for the file to be mirrored to the second/target Mac. On that second Mac, choose [Start Syncing Macros](#) from the [File menu](#). Again, Read the text carefully, and then click the [Open Existing](#) button.

WARNING: All of the existing macros on this second Mac will be destroyed if you continue. If you have any macros on the second Mac that you wish to preserve, export them first, and then after syncing is enabled, import them (and they will then be synced to your other Macs).

Select the mirrored sync file. Your existing macros will be replaced with the macros from your first Mac.

Repeat the process for any other Macs.

If there are some macros you do not want active on a Mac, you can configure any given Macro Group to be disabled on that particular Mac by turning on the [Disabled on this Mac](#) setting in that Macro Group.

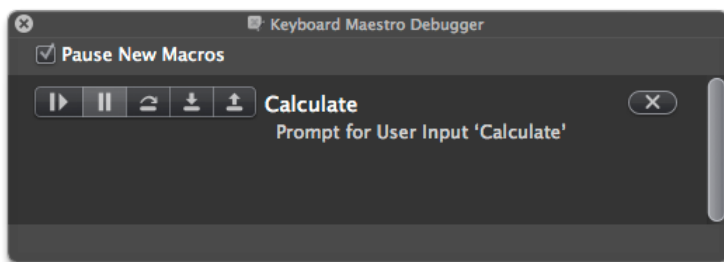
DropBox keeps backup versions, and Keyboard Maestro keeps backup revisions (in the [Revert Macros command](#) in the [File menu](#)), so you should be able to recover from any conflicts that happen. DropBox may notice a conflict if you edit your macros on both Macs simultaneously. As a general rule this should not be an issue, though you may lose a change if you make changes on both Macs quickly (and note that quitting the Keyboard Maestro editor is considered a change).

Since your two Macs will likely not be identical, you may have to adjust your macros to work properly on both Macs. Using the various [Text Tokens](#) and [functions](#) can help. For example the [%MacUUID%](#) is a unique ID for each Mac, and can be used to test which Mac the macro is running on. Other tokens, like the [%SCREEN%](#) token can be used to ensure your macro behaves appropriately regardless of the details of the Mac.

Note: Only your macros are synced. None of your preferences, clipboards or variables are synced.

Macro Debugger

Keyboard Maestro includes a built-in macro debugger which you can turn on by choosing [Start Debugging](#) from the [Status Menu menu](#) or by using one of the [Debugger Actions](#). This will display the [Macro Debugger window](#).



Once debugging, you can control whether new macros start paused or start running.

The macro debugger shows all running macros, and what action they are currently executing (including showing nested actions).

Using the associated buttons, you can:

- Continue the macro – allowing it to run until completion, or until it hits a Debugger Breakpoint action.
- Pause the macro.
- Step Over the current action, including any subactions.
- Step In To the current action, stepping in to any included subaction.
- Step Out Of the current action and any other actions at the same level.
- Cancel the macro.

All the macro actions can also be done via [Debugger Actions](#).

Keep in mind that once a macro has started executing, the engine has taken a copy of the macro to execute, so any changes you make in the editor will not affect the execution of the macro (although any changes you make to a sub macro that has not yet started executing would apply).

Also keep in mind that macros can often be time sensitive, so if you find your macro runs fine when stepped through in the debugger, but not when run normally, the issue is probably that the macro is executing actions like click actions before the system has caught up and the screen is stable. Add an appropriate Pause action if that is the case.

If you close the debugger window, the paused macros will resume their normal operation.

Variables

Keyboard Maestro includes permanently stored variables that you can use in a number of [special purpose actions](#), as well as in [Calculations](#) or [Text Tokens](#) which can be used in almost any field in Keyboard Maestro.

Variable names must start with a letter, and then can contain letters, numbers, spaces, or underscores. Variable names are case insensitive, but their case is remembered.

Variables with names that start or end with “Password” or “PW” are considered passwords – their values will not be stored (except in memory) and they cannot be read by shell scripts or AppleScripts. The Prompt For User Input dialog will display such variables in a password field.

Variable values are text, but they can contain comma separated numbers, and can then be accessed as arrays (eg Variable Name[1]). Mouse positions, window frames and the like can then be stored and manipulated in variables.

Variables values can be accessed from shell scripts via environment variables, and from AppleScript via environment variables or using AppleScript commands to the Keyboard Maestro Engine, and from JavaScript using the document.kmvar dictionary, see the [Scripting](#) section.

You can also set a variable value to `%Delete%` to avoid having it show up in variable popup menus.

There are various variables used by Keyboard Maestro to convey information, including:

Alert Button

the name of the button clicked in an alert.

Result Button

the name of the button clicked in a Prompt For User Input dialog.

Note that these variables can easily be overwritten by other actions, so you need to use caution when depending on them.

The following special variables were:

Mounted Volume Name

replaced by the %TriggerValue% token.

Mouse Click Result

replaced by the %ActionResult% token.

Press Button Result

replaced by the %ActionResult% token.

Select Menu Result

replaced by the %ActionResult% token.

USB Device Name

replaced by the %TriggerValue% token.

You can add, delete, see or change variables in the [Variables preference pane](#).

Filters

Keyboard Maestro includes a variety of filters that can be applied to either the system clipboard, [Named Clipboards](#), or variables. For clipboards, the filters preserve style information to whatever degree is possible. Use either the [Filter Variable action in the Variables category](#) or [Filter Clipboard action in the Clipboard category](#) to apply filters.

The filters are:

- Remove Styles (ie, make the clipboard plain text – not applicable to variables).
- Set line endings to Mac, Unix or Windows/DOS.
- Trim Whitespace.
- Hard wrap or unwrap paragraphs.

- Lowercase (all characters), Lowercase First (just the first character).
- Uppercase (all characters), Uppercase First (just the first character).
- Capitalize (all words) or Title Case (intelligently uppercase certain first letters).
- Change quotes to Smart, Dumb or French quotation marks.
- Encode HTML or non-ASCII HTML entities.
- Decode HTML entities.
- Generate an HTML list.
- Percent Encode for URL.
- Get or delete the last path component or the path extension.
- Get the basename of the path (ie the name without directory or extension).
- Expand tilde (~) paths, or abbreviate with a tilde.
- Resolve symlinks, or standardize the path.
- Delete or bullet (*) control characters.
- Calculate an expression and return the result, see the [Calculations](#) section.
- Process Text Tokens and return the result, see the [Text Tokens](#) section.
- Count the characters, words or lines and return the result.

We will likely expand the list of possible filters, so if you have specific filtering needs that you think might be of general interest, please let us know. In the mean time, remember that you can apply scripted filters using an AppleScript or shell script, for example the shell script:

```
pbpaste | perl -pe 'tr/A-Z/a-z/' | pbcopy
```

is roughly equivalent to the Lowercase filter, except that it only works with ASCII characters.

Text Tokens

Text Tokens allow you to add information to text fields. You can insert text tokens into a field using the Insert Token popup near the field.

The available text tokens include the following:

%Variable%Variable Name%

the value of the variable, see the [Variables](#) section.

%Calculate%1+2%

perform a calculation and use the result, see the [Calculations](#) section.

%CurrentClipboard%

the text of the current clipboard.

%PastClipboard%1%

the text of a past clipboard.

%NamedClipboard%Clipboard Name%

the text of a named clipboard.

%LongDate%

the current date in long format.

%ShortDate%

the current date in short format.

%NumberDate%

the current date in numeric format.

%LongTime%

the current time with seconds.

%ShortTime%

the current time without seconds.

%ICUDateTime%EEE, MMM d, yyyy%

the current date and time in any [ICU date format](#).

%ICUDateTimePlus%3*6%Hours%EEE, MMM d, yyyy%

an offseted date and time in any [ICU date format](#). Offset may be in Seconds, Minutes, Hours, Days, Weeks, Months, Years.

%ICUDateTimeMinus%3*6%Hours%EEE, MMM d, yyyy%

same as ICUDateTimePlus, but subtracts the offset.

%ICUDateTimeFor%NOW()+20%EEE, MMM d, yyyy%

a specific date and time in any [ICU date format](#).

%UserName%

the current user name.

%UserLoginID%

the current user login id.

%MacName%

the current Mac name.

%MacIPAddress%

the current Mac IP Address.

%MacUUID%

a unique ID for this Mac.

`%AddressBook%Name%`

your AddressBook name.

`%AddressBook%First%`

your AddressBook first name.

`%AddressBook%Last%`

your AddressBook last name.

`%AddressBook%Nickname%`

your AddressBook nickname.

`%AddressBook%Organization%`

your AddressBook organization.

`%AddressBook%Note%`

your AddressBook note.

`%CurrentApplication%`

the name of the current application.

`%LastApplication%`

the name of the last application.

`%Application%n%`

the name of the nth application.

`%FrontWindowPosition%`

the position of the front window.

`%FrontWindowSize%`

the size of the front window.

`%FrontWindowFrame%`

the frame of the front window.

`%FrontWindowName%`

the name of the front window.

`%WindowPosition%n%`

the position of the nth window.

`%WindowSize%n%`

the size of the nth window.

`%WindowFrame%n%`

the frame of the nth window.

`%WindowName%n%`

the name of the nth window.

`%CurrentMouse%`

the current mouse location.

`%Screen%Main%`

the frame of the main screen.

`%Screen%Second%`

the frame of the second screen (first from the left excluding the Main one).

`%Screen%Third%`

the frame of the third screen (second from the left excluding the Main one).

`%Screen%Internal%`

the (left-most) built-in screen.

`%Screen%External%`

the (left-most) non-built-in screen.

`%Screen%Front%`

the (left-most) screen containing (the most of) the front window.

`%Screen%Mouse%`

the (left-most) screen containing the mouse.

`%Screen%2%`

the frame of the second screen.

`%SystemVolume%`

the current system volume.

`%NetworkLocation%`

the current network location name.

`%NetworkLocation%`

the current network location name.

`%FindPasteboard%`

the current system find pasteboard value.

`%CurrentTrack%name%`

the current iTunes track name.

`%CurrentTrack%artist%`

the current iTunes track artist.

%CurrentTrack%album%

the current iTunes track album.

%CurrentTrack%ratingstars%

the current iTunes song rating.

%SafariTitle%

the title of the current Safari tab.

%SafariURL%

the URL of the current Safari tab.

%SafariReadyState%

the ready state of the current Safari tab.

%SafariField%

the value of a field in the current Safari tab.

%SafariJavaScript%

the result of some JavaScript executed in the current Safari tab.

%ChromeTitle%

the title of the current Google Chrome tab.

%ChromeURL%

the URL of the current Google Chrome tab.

%ChromeReadyState%

the ready state of the current Google Chrome tab.

%ChromeField%

the value of a field in the current Google Chrome tab.

%ChromeJavaScript%

the result of some JavaScript executed in the current Google Chrome tab.

%MailRecipients%

comma separated list of the recipients in the currently selected mail message.

%MailToRecipients%

comma separated list of the to recipients in the currently selected mail message.

%MailCCRecipients%

comma separated list of the cc recipients in the currently selected mail message.

%MailBCCRecipients%

comma separated list of the bcc recipients in the currently selected mail message.

%MailSender%

the sender of the currently selected mail message.

%MailReplyTo%

the reply to of the currently selected mail message.

%MailSubject%

the subject of the currently selected mail message.

%MailContents%

the contents of the currently selected mail message.

%MailRawSource%

the raw source of the currently selected mail message.

%ExecutingMacro%

the name of the currently executing macro.

%ExecutingMacroUUID%

the unique ID of the currently executing macro.

%ExecutingMacroGroup%

the name of the macro group containing the currently executing macro.

%Trigger%

the trigger that fired this macro.

%TriggerBase%

the trigger type (eg "Typed String Trigger")

%TriggerValue%

the value associated with the trigger (eg the typed string, or the hot key pressed).

%ActionResult%

the success or failure of the immediate past action.

%I%

(that is a vertical bar) allows you to position the cursor after insertion.

%Tab%

the tab (0x08) character.

%Space%

the space character.

%LineFeed%

the linefeed (0x0A) character.

%Return%

the carriage return (0x0D) character.

%NN% or %NNNN%

arbitrary hex unicode characters (eg %41% is an A).

\a,\b,\e,\f,\t,\r,\n

characters (bell,backspace,escape,form feed,tab,return,line feed).

You can also use a short form of just %Variable Name% to include variables as long as there is no corresponding text token.

The index for windows and applications is from front to back when positive, or from back to front when negative.

To include a percent in your text, simply double the percent (%%). To include a backslash \ in your text, double the backslash (\\).

Calculations

Keyboard Maestro supports calculations in almost any numeric field. For example you can Pause for 60*Time in Minutes. Calculations can also use comma separated lists of numbers as arrays, and can return such arrays, so you can operate on frames and points, for example:

```
Set variable 'Temp' To Text '%FrontWindowFrame%'
Set variable 'Temp' To Calculation 'Temp[1]+Temp[3]/2,Temp[2]+Temp[4]/2'
Use 'Temp' to Set the Mouse Location
```

will result in the mouse being placed at the center of the front window.

Unfortunately, because of this you must use commas for this purpose, and full stops (.) for decimal numbers, regardless of your desired language, and never use any thousands separators.

Keyboard Maestro's expressions include precedence, nested bracketed expressions, many built-in functions, various numeric bases, so you should be able to write most expressions you might like to use, as well as use it as a general purpose calculator if desired.

Operators based on precedence from lowest to highest are:

array separator (,)

separates elements of an array.

ternary operator (?)

a=b ? 3 : 4.

bitwise or (|), bitwise and (&) and bitwise xor

operators.

comparison operators (<, ≤, =, >, ≥, ≠)

compare for (in)equality and return 0 or 1.

shift operators (<<, >>)

shift a number left or right.

addition operators (+, -)

add or subtract.

multiplication operators (*, /, MOD)

multiply, divide or mod.

power operator (^)

exponentiation.

unary prefix operators (√, -, brackets)

square root, negation, sub-expressions.

functions

a variety of functions.

numbers and variables or array accesses (5,\$5A,0x50,8#007,Variable,Variable[5])

identifiers and values.

unary postfix operators (!,%°,)

factorial, percent, degrees.

To minimize conflict with variables, operators and functions must be in uppercase.

Supported functions include:

SIN, COS, TAN, ASIN, ACOS, ATAN, ATAN2

trigonometry functions

[SINH](#), [COSH](#), [TANH](#), [ASINH](#), [ACOSH](#), [ATANH](#)

hyperbolic trigonometry functions

[EXP](#)

exponentiation of e.

[LOG](#) or [LN](#)

logarithm base e.

[LOG2](#), [LOG10](#)

logarithm base 2 or 10.

[ABS](#)

absolute value.

[CEIL](#), [FLOOR](#)

integer ceiling or floor.

[TRUNC](#), [ROUND](#)

truncate or round.

[MIN](#), [MAX](#)

Minimum or maximum.

[RANDOM](#)

random real number from 0 to, but not including, N (defaults to 1).

[RAND](#)

random integer number from 0 to, but not including, N.

[MICROSECONDS](#) or [MICROS](#)

microseconds since startup.

[MILLISECONDS](#) or [MILLS](#) or [MS](#)

fractional milliseconds since startup.

[SECONDS](#)

fractional seconds since startup.

[NOW](#)

current time in seconds since the start of 1970.

[TRIGGERTIME](#)

the time in (fractional) seconds since the start of 1970 when the macro was triggered.

[IDLE](#)

the number of (fractional) seconds the Mac has been idle.

[TIME](#)

current time in seconds since the start of 1970.

[JULIANDATE](#) or [JD](#)

days (and fractions of days) since January 1, 4713 BC Greenwich noon.

[MJD](#)

days since the start of November 17, 1858.

[TIME2JD](#)

convert time from unix TIME to JULIANDATE.

[TIME2MJD](#)

convert time from unix TIME to MJD.

[JD2MJD](#)

convert time from JULIANDATE to MJD.

[JD2TIME](#)

convert time from JULIANDATE to TIME.

[MJD2TIME](#)

convert time from MJD to TIME.

[MJD2JD](#)

convert time from MJD to JULIANDATE.

[YEAR](#)

get the year component from a unix time (seconds since the start of 1970).

[MONTH](#)

get the month component from a unix time.

[DAY](#)

get the day component from a unix time.

[DOW](#)

get the day of the week (1 = Sunday, 7=Saturday) component from a unix time.

[HOUR](#)

get the hour component from a unix time.

[MINUTE](#)

get the minute component from a unix time.

SECOND

get the second component from a unix time.

GMTOFFSET

get the number of seconds from GMT.

MOUSEX

the x coordinate of the current mouse location.

MOUSEY

the y coordinate of the current mouse location.

MOUSEBUTTON

1 if the corresponding button is pressed, 0 otherwise.

SCREEN(n, Left|Right|Top|Bottom|Width|Height|MidX|MidY, p)

screen frame coordinates.

WINDOW(n, Left|Right|Top|Bottom|Width|Height|MidX|MidY)

window frame coordinates.

SCREENCOUNT

the number of screens (displays).

WINDOWCOUNT

the number of windows in the front application.

APPLICATIONS

the number of running applications.

SAFARITABCOUNT

the number of tabs in the front Safari window.

SAFARITABINDEX

the index of the selected tab in the front Safari window.

SAFARIISCOMPLETE

whether Safari has currently finished loading the current tab.

CHROMETABCOUNT

the number of tabs in the front Google Chrome window.

CHROMETABINDEX

the index of the selected tab in the front Google Chrome window.

CHROMEISCOMPLETE

whether Google Chrome has currently finished loading the current tab.

MAILDATERECEIVED

the date received of the currently selected mail message.

MAILDATESENT

the date sent of the currently selected mail message.

MAILFLAG

the flag of the currently selected mail message.

MAILFLAGGED

the flagged status of the currently selected mail message.

MAILREADSTATUS

the read status of the currently selected mail message.

MAILJUNKSTATUS

the junk status of the currently selected mail message.

MAILWASFORWARDED

the was forwarded status of the currently selected mail message.

MAILWASREDIRECTED

the was redirected status of the currently selected mail message.

MAILWASREPLIEDTO

the was replied to status of the currently selected mail message.

ONLINE

whether the Mac is currently connected to the Internet.

BATTERY

whether the Mac is currently running off battery power.

SCREENSAVER

whether the Mac is currently displaying the screen saver (or the display is off).

SYSTEMVOLUME

current system sound output volume (0-100).

CLIPBOARDSEED

an integer that changes when the clipboard changes.

The JD, MJD, and TIME functions return the current date and time, or can take the date (year, month, day) or date and time (year, month, day, hour, minute, seconds) to return.

The YEAR, MONTH, DAY, DOW, HOUR, MINUTE, SECOND functions return the relevant component of the

current time or the can take the a unix time (seconds since the start of 1970).

The SCREEN index can be 0 for the main screen, and then 1 through n are the screens in orientation order from left to right. The index can also be one of:

Main

the screen with the menu bar.

Second

the left most screen that does not have the menu bar.

Third

the left most screen that is not the Main or Second screen.

Internal

the left most screen that is an internal screen (typically on a laptop).

External

the left most screen that is not an internal screen.

Mouse

the left most screen containing the mouse.

Front

the left most screen containing (the most of) the front window.

Back

the left most screen not containing (the most of) the front window.

Back2

the second left most screen not containing (the most of) the front window.

The optional p parameter specifies a percentage of the width or height to offset by. Eg SCREEN(Internal,Left,10%) would be the coordinate of left edge of the internal screen plus 10% of the width of the internal screen. Offsets are always to the right and down, but negative offsets are allowed.

The WINDOW index can be 0 for the main focussed window, and 1 through n are the screens in Z-order (1 is usually the main window), or -1 through -n in reverse order.

IDLE time is based on the Human Interface (HID) system, and so notices only HID device activity like mouse movement or keyboard presses, not things like disk access or movies playing.

Some actions have one or two images, or refer to a window or some text, and calculations in those actions can include reference to some context sensitive functions:

IMAGE(Width|Height)

the action image size.

SOURCEIMAGE(Width|Height)

the action source image size.

WINDOW(Left|Right|Top|Bottom|Width|Height|MidX|MidY)

the action window coordinates.

LENGTH()

the action text length.

FONTSIZE()

the original font size during the Apply Style action.

Keyboard Maestro refers to points as strings with two values, like 12,34 and to rectangles as four values 12,34,56,78. You can reference these values as arrays (eg Variable[2] would be 34 in either case), but for clarity, you can reference these values using dot notation:

Variable.x

x coordinate.

Variable.y

y coordinate.

Variable.left

the left coordinate of a rectangle.

Variable.top

the top coordinate of a rectangle.

Variable.right

the right coordinate of a rectangle.

Variable.bottom

the bottom coordinate of a rectangle.

Variable.width

the width of a rectangle.

Variable.height

the height of a rectangle.

Variable.MidX

the horizontal middle of a rectangle.

Variable.MidY

the vertical middle of a rectangle.

Some example functions might be:

```
Amount in Dollars * 100
MJD() > 55928
NOW() > TIME(2012,3,23,12,2,1)
DOW(TIME(2012,4,4)) = 4
Radius*SIN(20°),Radius*COS(20°)
Window Frame[1]+Window Frame[3]/2,Window Frame[2]+Window Frame[4]/2
MOUSEBUTTON() + 2 * MOUSEBUTTON(4)
SCREEN(Internal,Left,10%)
```

Conditions

Keyboard Maestro includes a variety of [Control Flow Actions](#) which perform actions depending on a set of conditions.

The condition clause of the flow control actions can be any of:

- Any of the following are true – at least one condition must be true.
- All of the following are true – every condition must be true.
- None of the following are true – no condition is true.
- Not all of the following are true – at least one condition must be false.

This is followed by a set of specific conditions. **If there are no conditions in the set at all, the action will not execute anything** except the Until action which will execute the actions once. Neither side of the If Then Else will execute.

The available conditions include:

Application

test if an application is running or at the front (or not).

Front Window

test if the front window exists or has a desired title (or not).

Button

test if a button exists, is enabled or is checked (or not).

Menu

test if a menu item exists, is enabled or is marked (or not).

Modifiers

test if particular modifiers are pressed or not pressed.

Key

test if a particular key is down or up.

Typed String

test if a typed string trigger with remember case was a particular case.

Mounted Volume

test if a disk is mounted or not.

Path

test if something, a file or folder exists at a particular path (or not).

Clipboard

test if the clipboard contains text or an image or specific text (or not).

Variable

test if a variable exists or is empty or contains specific text (or not).

Text

test if a piece of arbitrary tokenised text is empty or contains specific text (or not).

Calculation

test if a calculation returns non zero.

Environment Variable

test if an environment variable exists or contains specific text (or not).

Screen Image

test if the screen contains (optionally uniquely) an image (or not).

Pixel

test if a pixel on the screen is or is brighter/darker, more or less blue, etc that a particular color.

Location

test if the network name is or contains specific text (or not).

USB Device

test if a USB device with a specified name exists (or not).

Wireless Network

test if a Wireless Network with a specified name is connected (or not).

Script

test if a script succeeds or returns particular status or specific text (or not).

Generally, a screen image and pixel conditions should be considered a last resort, but there are certain cases where it may be useful – keep in mind that you can use expressions for calculating the location of the pixel too.

When talking about matches, note that “is” refers to equality, “contains” looks for a substring, “matches” uses regular expression matching, and “conforms” refers to Universal Type Identifier matching (eg `public.text`).

See also the Control Flow Actions, Variables and Calculations sections.

Recording

Keyboard Maestro has the ability to create macro action sequences by recording your actions.

For example, to create a macro that simulates keystrokes, rather than create each macro action individually you can enable recording and then simply type the keystrokes.

Keyboard Maestro can record the following actions:

- Moving a window
- Resizing a window
- Miniaturizing a window
- Clicking the mouse
- Typing a Keystroke
- Moving the scroll wheel
- Selecting a menu
- Activating an application
- Quitting an application

There are two ways you can use recording: when creating or editing a macro, or via a Record Quick Macro action.

When you are creating or editing a macro, with the Macro Editor window displayed, simply click the **Record** button. After a short pause for you to get ready, recording will begin. To avoid the pause, hold the option key down while clicking the **Record** button).

Once recording starts, demonstrate the task you would like to perform using any of the above actions and Keyboard Maestro will record your actions directly into your macro.

While Keyboard Maestro is recording, it will display the Recording window.



While you are recording, you can pause the recording by clicking the **Pause** button in the recording window, and you can add a 1 second pause to your macro by clicking the **Clock** button.

When you are finished, click the **Record** button again to stop recording, or you can stop all recording by clicking on the Recording window.

Typically you will need to make a few adjustments to the Macro Actions to ensure the macro will operate robustly when used. Generally, use recording to create a base sequence of actions and then adjust as necessary.

The other way to use recording is via a Record Quick Macro action. When triggered, the Record Quick Macro immediately starts recording your actions into a private macro. When you have demonstrated the sequence of actions you want, trigger the Record Quick Macro action again. The sequence can now be executed via the specified Hot Key or the Status Menu or Macro Palette. For example, if the Record Quick Macro is triggered by pressing Control-F1, and the specified Hot Key is Option-F1, then if you typed:

Control-F1, h, e, l, l, o, Control-F1

Then each time you press Option-F1, Keyboard Maestro will type “hello” for you. One common use for this is if you want to adjust a sequence of lines in a systematic way. For example, if you had a list of colors, and

wanted to change them in to a list of constants, say from this:

```
color Red
color Green
color Blue
```

to

```
const int kRed = "Red";
const int kGreen = "Green";
const int kBlue = "kBlue";
```

You could do this with [grep](#) and [regular expression](#), replacing “color (.*)” with “const int k\1 = “\1”;;”, which is fine if you can remember how to do [grep with regular expressions](#), whether it is \1 or \$1, and whether the application you are in supports [regular expressions](#) or not. But perhaps a simpler way is to just show Keyboard Maestro how to do the first line and then let it do the others with a single keystroke each.

So move the cursor to the start of the first line, press Control-F1, then the sequence:

Option-Shift-Right Arrow, Delete, Forward Delete, Command-Shift-Right Arrow, Command-X, c, o, n, s, t, <space>, i, n, t, <space>, k, Command-V, =, ", Command-V, ", ;, .

Finish with Command-Left Arrow, Down Arrow to carefully put the cursor at the start of the next line. Now press Control-F1 again to finish the recording, and Option-F1 twice to translate the next two lines.

Record Quick Macros can record the same set of actions that normal recording can, however because you cannot see or edit the recorded actions it is wise to keep them simple, preferably just a sequence of keystrokes. Typically, recorded Quick Macros will be used immediately and not reused, but they are saved and remain available until you record over them.

Macro Library

The macro library is a place where we can provide you with a variety of ready-made macros for optional addition to your macros. You can download new potential macros from us or from friends or colleagues. You can also share your own macros with other Keyboard Maestro users by exporting your clever macros as a library item.

Keep in mind that macros can do practically anything on your Mac, including cause a huge amount of damage, so you should never execute a macro without verifying the source and better yet, checking exactly what it does.

To use the library, choose [Macro Library](#) from the [Window menu](#) to display the macro library. You can then look through the available macros and insert any you'd like to use into your macros. You can then use them as is, or configure the new macros, perhaps changing the hot keys or adjusting the macros to your liking.

Each macro comes with a short description to tell you what it does, so scroll through them to see all the possibilities, and click on them to get more details.

You can export a set of macros to a macro library file, which you can share with others, and you can import .kmlibrary files into your macro library. Note that currently there is no way to delete imported macro library entires from your Macro Library except by quitting Keyboard Maestro and Keyboard Maestro Engine and removing the files from the `~/Library/Application Support/Keyboard Maestro/Keyboard Maestro Libraries` folder

Macro Examples

Here are a number of example and suggestions for Macros to give you some ideas of how you can get the most out of Keyboard Maestro and your Mac. For tips on how to remember which [Hot Key](#) executes which action, see the [Remembering Macro Hot Keys](#) section.

- [Launch Your Most Used Applications](#)
- [Open Your Most Used Documents](#)
- [Insert Text Templates](#)
- [Use Hot Keys to Open Financial Accounts](#)
- [Use Hot Keys to Connect to SSH or FTP sites](#)
- [Simulate Bookmarks](#)
- [Remap Command Keys](#)
- [Simulate Missing Features](#)
- [Swap Characters](#)
- [Save a Text Clipping](#)
- [Delayed Click](#)
- [Insert Boilerplate Text](#)
- [Apply Text Conversions](#)
- [Simulate Workspaces](#)

- [Setup an Application When Launched](#)
- [Clean Up After Using an Application](#)
- [Launch Scanner Application When Scanner is Connected](#)
- [Switch Network Location When You Connect](#)
- [Feedback During Macro Execution](#)
- [Rakesh Kumar's PC Switcher's Pack](#)

Launch Your Most Used Applications

Use function keys to launch or switch to your most used applications. For example, you probably often switch to the Finder, your Email client, your Web Browser, your Word Processor. Consider putting these and other frequently used applications on function keys.

Open Your Most Used Documents

Use Control-Function Keys to open your most used documents. For example, you might have a documentation file or financial details file that you access frequently, consider putting these on Control-Function Keys.

Insert Text Templates

Use Control-Letter and the [Insert Text](#) action to type in text for you, such as your name, address, phone number, and so on. Consider restricting these to just the appropriate applications like your Email client or Word Processor by creating a [Macro Group](#) for them. Also consider using Typed String triggers for these sorts of macros, for example “=em=” for email address and “=addr=” for address. The text you insert can be typed, pasted as plain text, or can be fully styled text.

Use Hot Keys to Open Financial Accounts

If you keep your finances on your computer, then you probably need to open a document every time you enter a bill or receive a statement. By creating a [Hot Key](#) to open the document for you, you can save a few seconds every time – at least it might make receiving a bill slightly less unpleasant! If you have multiple accounts (eg personal, business, association) then this can be even more useful.

Use Hot Keys to Connect to SSH or FTP sites

You could use [Hot Keys](#) to connect to your common servers. You might need to create a Bookmark file for the site and use the Open File [Macro Action](#).

Simulate Bookmarks

Use the [Safari and Google Chrome Actions](#) to open web pages, fill in fields, submit forms, follow links. For example, you could use this to log in to the city library for all the members of your family, one in each tab, to easily check what books are due back.

If you are going to use this to enter passwords, use the Set Variable to Keychain Password action to retrieve the password so that it is not stored in plain text in the Keyboard Maestro macros.

Remap Command Keys

If you find yourself pressing a command key in an application and expecting it to do something but it does not work (for example, Command-T for “Replace and Find Again”), use a Macro to make the command key “do the right thing” in that application. Similarly, if you use a function in an application frequently, but it has a convoluted command key or no command key at all, define your own command key by using a [Hot Key](#) to select the menu item.

Keep in mind that you can do some menu key remapping in the System Preferences Keyboard preference.

Simulate Missing Features

If you find yourself missing a feature in one application that you are used to in another application (perhaps you switched email clients and a feature is missing), see if you can simulate the feature with a sequence of commands and then use a [Hot Key](#) for that. For example, Close Window, Down Arrow, Return to move to

next email message, or Command-Left Arrow, Shift-Down Arrow, Command-C, Down Arrow, Command-V to duplicate a line.

Swap Characters

If you often type characters out of order, use a Hot Key to swap them by first placing the cursor between them and then executing:

- Simulate Keystroke Shift-Right Arrow
- Cut to Named Clipboard “Temp”
- Simulate Keystroke Left Arrow
- Paste from Named Clipboard “Temp”

Save a Text Clipping

If you often want to save snippets of text, you could create a Hot Key to save a clipping:

- Copy
- Open File “Clippings.rtf”
- Simulate Keystroke Command-Down Arrow
- Insert Text “== %LongDate% %ShortTime% ==<return>” by Typing
- Paste
- Simulate Keystroke Return
- Simulate Keystroke Return
- Select Menu Item File » Save
- Manipulate Window Close Front Window
- Switch to Last Application (or Quit Specific Application or Command-Q)

Delayed Click

Setup a macro which simply pauses for twenty seconds and then clicks the mouse. Then when you need to print on to an envelope, go all the way through the process, position the mouse over the Print button, execute the Macro, walk over to the printer, insert an envelope and then take the printed envelope back with you.

Insert Boilerplate Text

If you regularly need to insert boilerplate text (eg copyright or file creation text), use an Insert Text macro to insert the text quickly and easily. It can even expand tokens to insert the date or other information.

Apply Text Conversions

If you are regularly translating text from one format to another in an automatic process, perhaps you can automate the whole thing with a Keyboard Maestro macro. For example, converting function declaration in a header file into function definition.

Simulate Workspaces

Create a macro to setup an application to your liking. For example, create multiple tabs in Terminal, each in its own directory, or open multiple documents in TextEdit, each positioned and sized appropriately.

Setup an Application When Launched

If you always do a set of things every time you launch an application (eg arrange the windows in a particular way), use an application Macro Trigger to execute a Macro when you launch the application, then have the Macro do the work for you.

Clean Up After Using an Application

If you always do something after quitting an application (eg unmount a server or disconnect from the Internet), use an application Macro Trigger to execute a Macro when you quit the application. You might need to do a little AppleScripting to perform the action and then use the Execute AppleScript action.

Launch Scanner Application When Scanner is Connected

Set up a macro that automatically launches your scanner application when your scanner is connected, and quits it again when the scanner is disconnected. This works brilliantly with the ScanSnap scanners – open the lid and the scanner software launches, close it and the scanner software disappears.

Switch Network Location When You Connect

Set up a macro that automatically changes your Network Location when you connect to your home or work wireless network.

Feedback During Macro Execution

A Macro can play a System Beep, but an alternative is to use an AppleScript or shell script to speak text (AppleScript `say "hello"`).

You can also use the Alert action to display a window with specified text. This also allows you to stop the macro if you decide not to proceed.

Rakesh Kumar's PC Switcher's Pack

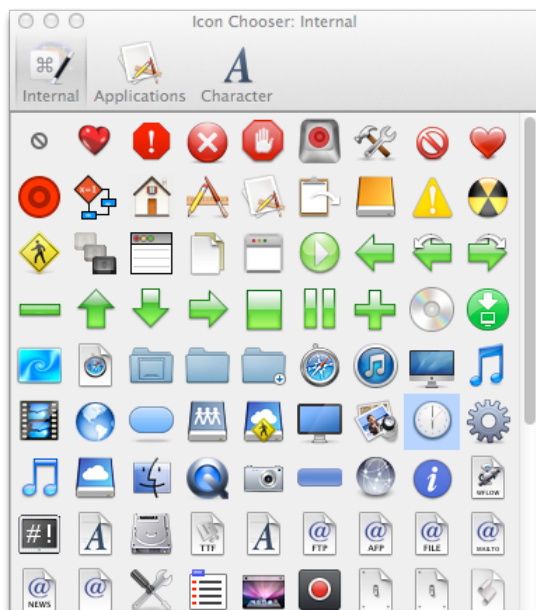
Rakesh Kumar has created a pack to make life easier for Windows to Mac Switchers. It includes a set of macros for Mail, Microsoft Word and Microsoft PowerPoint as well as macros to map control keys to command keys for various common actions like Cut/Copy/Paste. It also includes a DefaultKeyBinding.dict for Mail to make it work more like Windows users expect.

Download [Rakesh Kumar's PC Switcher Pack](#) and follow the instructions.

Icon Chooser

Keyboard Maestro includes an Icon Chooser and creator to allow you to select custom icons for your macro groups and macros.

You can display the Icon Chooser by choosing [Icon Chooser](#) from the [Window menu](#).

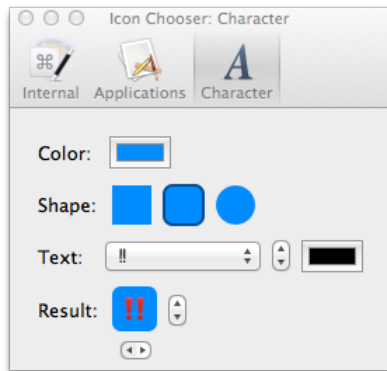


Click on an image well for a macro or macro group (in Edit mode), and then click on an icon in the Icon Chooser to select it.

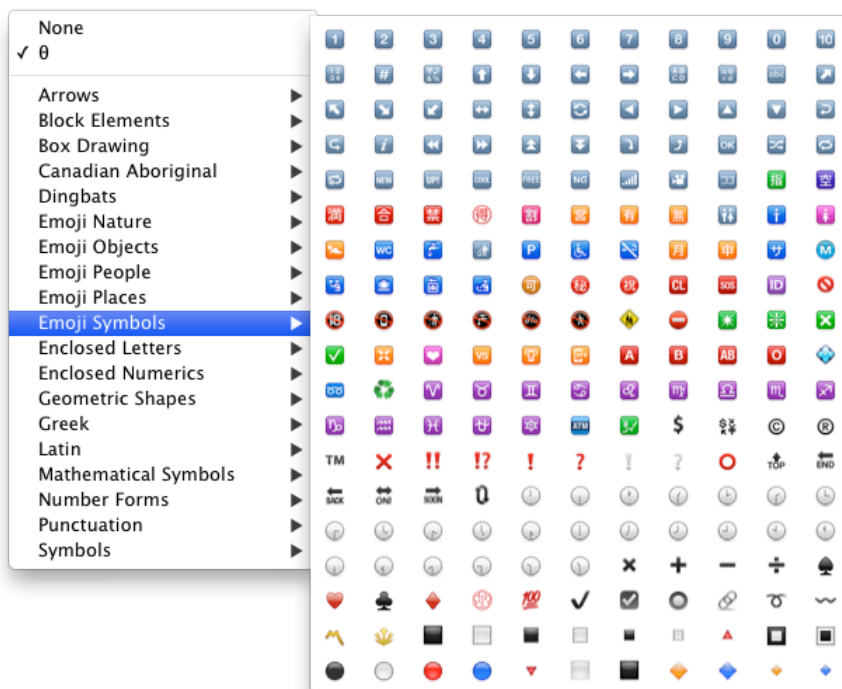
The Icon Chooser includes three panes with different icons:

- Internal – all the icons Keyboard Maestro normally uses.
- Applications – all the icons Keyboard Maestro can find on your Mac.
- Character – icons you create based on a character.

The [Character Icon Choose pane](#) allows you to create your own custom icons by setting a few simple parameters.



You can set the background and foreground color, shape, and an optional character:



Application Launcher

The Activate Application Launcher action is essentially a highly specialized macro action that enables you to launch applications. By triggering the macro, the Application Launcher enables you to launch any applications in your Applications or Utilities folder, as well as any recently running applications. Once the launching window appears, you may select the application to launch, and Application Launcher will launch it for you.

The applications are also listed in the Status Menu, so you can launch applications that way if you prefer.

By default, Keyboard Maestro creates a Activate Application Launcher macro in the “Switcher Group” Macro Group, triggered by Command-Control-Tab. You can disable this Macro by selecting the Switcher Group, then selecting the Activate Application Switcher macro and clicking the ☒ button below the Macros list. You can edit this macro to change the trigger to any other desired [Hot Key](#).

Application Switcher

The Activate Application Switcher action is essentially a highly specialized macro action that enables you to launch, switch, hide, and quit applications. By triggering the macro, the Application Switcher enables you to switch between all running applications. Once the switching window appears, you may select the application to activate, and Application Switcher will take you to it.

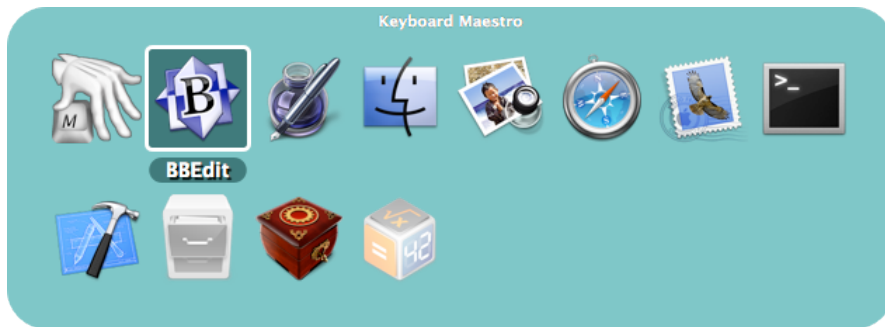
The Activate Application Switcher action lets you choose from three themes (vertical list, horizontal icons, or icon grid), as well as configure the icon size, color tint, and the sort order.

You can also choose to hide other applications when switching (Keyboard Maestro also has a preference in the [General preference pane](#) to always hide other applications when switching).

You can select various applications to always be displayed, even if they are not currently running, perfect for launching frequently used applications. In the [Excluded preference pane](#), you can configure various applications to never be displayed.

While the [Application Switcher window](#) is displayed, you can perform various actions:

- Press “q” to mark (or unmark) an application to be quit.
- Press “k” twice to mark an application to be force quit.
- Press “s” or “h” to mark (or unmark) an application to be hidden.
- Press “l” or “z” to mark (or unmark) an application to be launched.
- Press “a” to hide (or show) “always included” applications.
- Press “e” to show (or hide) “always ignored” applications.
- Press “c” to select the current application.
- Press “f” to select the Finder.
- Press “d” to switch directly to the current app and hide other applications.



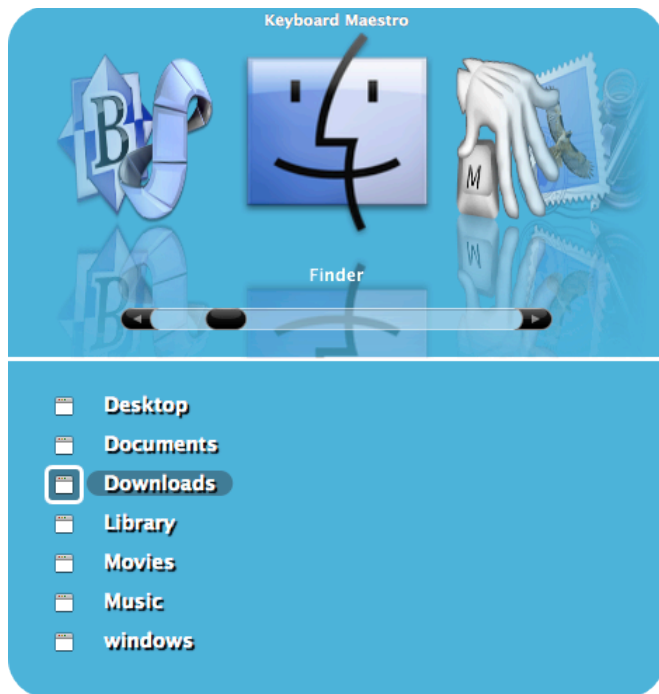
By default, Keyboard Maestro creates a Activate Application Switcher macro in the “Switcher Group” [Macro Group](#), triggered by Command-Tab. You can disable this Macro by selecting the Switcher Group, then selecting the Activate Application Switcher macro and clicking the ☒ button below the Macros list. You can edit this macro to change the trigger to to any other desired [Hot Key](#) avoid replacing the system application switcher.

Window Switcher

Activate Window Switcher is essentially a highly specialized macro action that enables you to show, hide, and minimize windows. By triggering the macro, Window Switcher enables you to switch between all open windows in the current application. Once the switching window appears, you may select the window to activate, and Window Switcher will bring it to the front.

While the [Window Switcher window](#) is displayed, you can perform various actions:

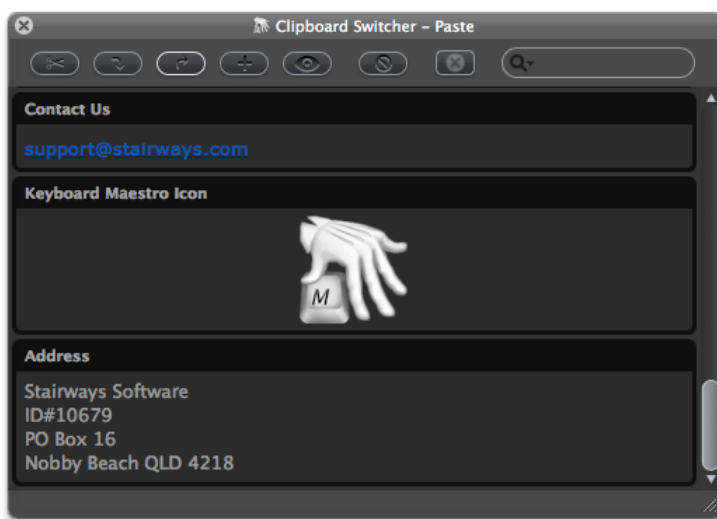
- Press “q” to mark (or unmark) a window to be closed.
- Press “s” to mark (or unmark) a window to be minimized.
- Press left or right arrow keys to scroll through applications.



By default, Keyboard Maestro creates a Activate Window Switcher macro in the “Switcher Group” Macro Group, triggered by Control-Tab. You can disable this Macro by selecting the Switcher Group, then selecting the Window Switcher macro and clicking the ☒ button below the Macros list.

Clipboard Switcher

Clipboard Switcher enables you to define any number of named clipboards which can be used to Cut or Copy into and Paste from in any application. To use Clipboard Switcher you simply trigger the Clipboard Switcher macro. Clipboard Switcher will present you with a window allowing you to select the named clipboard to use.



You can press arrow keys to scroll through the clipboard entries, or you can use type-ahead to select a named clipboard, and you can use the search field to filter the clipboards.

You can cut or copy the current selection into the selected clipboard, or paste the selected clipboard into the current selection (hold down the shift key to paste as plain text).

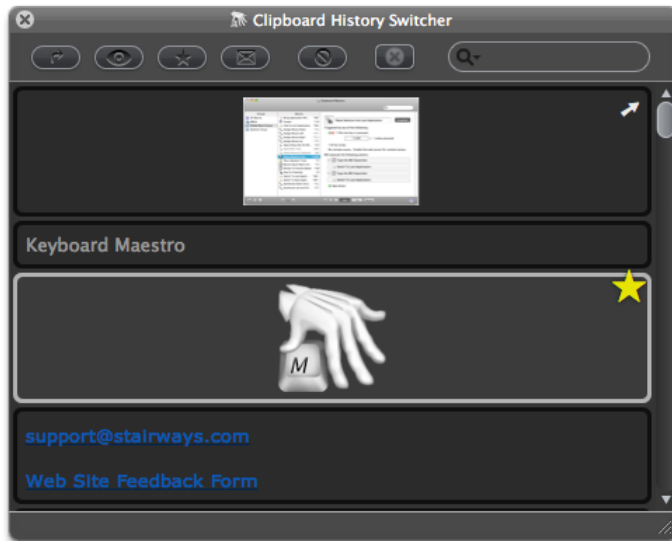
By default, Keyboard Maestro creates three Clipboard Switcher macros in the “Switcher Group” Macro Group, triggered by Command-Shift-X, C and V (Cut, Copy and Paste). You can disable the Macros by selecting the Switcher Group, then selecting the macros and clicking the ☒ button below the Macros list.

You can add or delete named clipboards in the Clipboards preference pane.

Clipboard History Switcher

Clipboard History Switcher saves a copy of each clipboard every time you copy something. You can then

paste any previous system clipboard by triggering the Clipboard History Switcher macro. Clipboard History Switcher will present you with a window allowing you to select any of the past clipboard and paste them.



You can press arrow keys to scroll through the clipboard entries, and return/enter to paste in an entry. Hold the Shift key while pressing return or double clicking to paste as plain text.

You can use the search field to filter the clipboards.

You can mark entries as favorites, in which case they will never be removed from the clipboard history.

Clipboard entries that resemble passwords are obscured, deleted after they reach position ten in the clipboard history, and not saved to disk. You can option click on an obscured password to reveal it.

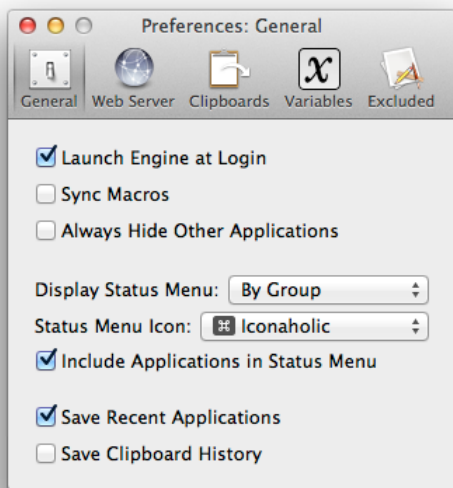
You can send clipboards to another Mac running Keyboard Maestro. The received clipboard entries will appear in the clipboard history (assuming the Keyboard Maestro Web Server is enabled on the destination Mac).

By default, Keyboard Maestro creates a Clipboard History Switcher macro in the “Switcher Group” Macro Group, triggered by Command-Control-Shift-V. You can disable the Macros by selecting them in the Macros window pane and clicking the ☒ button.

Keyboard Maestro also creates disabled macros for Paste Plain Text (Command-Shift-V) and Paste Previous Clipboard (Command-Control-V) in the “Global Macro Group” Macro Group. You can enable these by selecting the Global Macros Group, then selecting the macros and clicking the ☒ button below the Macros list.

Preferences

To configure Keyboard Maestro, first launch Keyboard Maestro and choose Preferences from the Keyboard Maestro menu.



The preferences are divided into sections.

General Preferences

In the General preference pane you can:

- enable or disable launching the Keyboard Maestro Engine at Login.
- configure whether to always hide other applications when switching applications.
- configure whether to always show the status menu.
- configure whether to detect clipboards that resemble passwords and obscure them.
- configure the style of the status menu.
- configure whether to include applications in the status menu.
- configure whether to save the recent applications between logins.
- configure whether to save the clipboard history between logins.

Web Server Preferences

In the Web Server preference pane you can:

- enable or disable the web server.
- configure the username, password and port of the web server.
- enable or disable web browser access.
- enable or disable iPhone access.
- enable or disable receiving clipboard entries.
- enable or disable replacing the current clipboard.
- access the web server in your default browser by clicking on Connect.

The web server is disabled by default.

If the web server and web browsing are enabled, then anyone who can connect to your Mac can execute any macro that has a Public Web trigger.

If the web server and web browsing are enabled, and if you have configured a username and password, then anyone who can connect to your Mac and login with the specified username and password can execute any of your macros.

If the web server and iPhone access are enabled, and if you have configured a username and password, then anyone who can connect to your Mac from an iPhone and login with the specified username and password can execute any of your macros.

If the web server and receiving clipboard are enabled, then anyone can send you clipboards which will appear in your clipboard history. By default they do not overwrite the current clipboard, but you can enable that to allow the current clipboard to be directly written remotely.

Macros are still only available when they are enabled and their containing macro group is enabled and active.

You can configure custom styles using the defaults write command to add a custom style, for example:

```
defaults write com.stairways.keyboardmaestro.editor WebServerCustomStyles -string 'body { bac
```

This might be useful if you are controlling multiple Macs and want to differentiate them more clearly.

Clipboards Preferences

In the [Clipboards preference pane](#) you can add, remove and rename [Named Clipboards](#) and see and change their values.

[Named Clipboards](#) store snippets or text or images (or anything the clipboard can hold) and you can copy or paste from them using the [Clipboard Switcher](#) or using appropriate macro actions.

You can paste an image into a [Named Clipboard](#) by selecting it in the list and pasting an image (if you select part of the text, it will paste the image into the styled text, which works but is probably not what you want).

Variables Preferences

In the [Variables preference pane](#) you can add and remove variables and see and change their values.

Excluded Preferences

In the [Excluded preference pane](#) you can add and remove applications from the global excluded applications list. Excluded applications will not be shown in the Application Switcher, and are (optionally) not hidden or quit by the Hide All Applications or Quit All Applications actions.

Other Hidden Preferences

You can configure various preferences using defaults write from the Mac OS X Terminal. Some preferences will take effect immediately, but others may require the engine to be relaunched.

You can set the maximum number of items in the clipboard history (default 100)

```
defaults write com.stairways.keyboardmaestro.engine MaxClipboardHistory -int 100
```

You can set the maximum position of concealed items in the clipboard history (default 10)

```
defaults write com.stairways.keyboardmaestro.engine MaxConcealedPosition -int 10
```

You can set a regular expression which matches text you think should or should not be concealed as passwords.

```
defaults write com.stairways.keyboardmaestro.engine LooksLikePassword "^[a-zA-Z0-9]+$"
defaults write com.stairways.keyboardmaestro.engine LooksLikeNotPassword "^[a-zA-Z0-9]+$"
```

You can set a delay between each action execution (default 0.0)

```
defaults write com.stairways.keyboardmaestro.engine InterActionDelay -float 0.5
```

You can set the delay to wait after pasting before continuing to the next action (default 0.2)

```
defaults write com.stairways.keyboardmaestro.engine ClipboardDelay -float 0.2
```

You can set the delay to wait after simulating a keystroke before continuing to the next action (default 0.0)

```
defaults write com.stairways.keyboardmaestro.engine SimulateKeystrokeDelay -float 0.01
```

If the command key is down, the delay is stored in the SimulateKeystrokeCommandKeyDelay preference (default 0.15)

```
defaults write com.stairways.keyboardmaestro.engine SimulateKeystrokeCommandKeyDelay -float 0.15
```

The delay between dead key and following key is stored in the SimulateKeystrokeDeadKeyDelay preference (default 0.03)

```
defaults write com.stairways.keyboardmaestro.engine SimulateKeystrokeDeadKeyDelay -float 0.03
```

You can set the command line tool that is used to execute AppleScripts

```
defaults write com.stairways.keyboardmaestro.engine OSAScriptCommand "/usr/bin/arch -i386 /usr/bin/osascript"
```

You can disable Shift-Space from clearing the Typed String buffer

```
defaults write com.stairways.keyboardmaestro.engine TypedStringResetWithShiftSpace -bool NO
```


and you can set the idle time for clearing the Typed String buffer (default 5 seconds)

```
defaults write com.stairways.keyboardmaestro.engine TypedStringResetWithShiftSpace -float 5.0
```

You can disable all animation in the editor with:

```
defaults write com.stairways.keyboardmaestro.editor DisableAnimation -bool YES
```

or the engine with:

```
defaults write com.stairways.keyboardmaestro.engine DisableAnimation -bool YES
```

You can silence the clipboard transfer sounds with:

```
defaults write com.stairways.keyboardmaestro.engine SilenceClipboardSounds -bool YES
```

And you can silence the recording sounds with:

```
defaults write com.stairways.keyboardmaestro.engine SilenceRecordingSounds -bool YES
```

You can adjust the recording delay with:

```
defaults write com.stairways.keyboardmaestro.engine RecordingCountDown -int 5
```

Note: you can option click the Record button to avoid the delay.

You can adjust the spelling of Favorites with:

```
defaults write com.stairways.keyboardmaestro.editor FavoritesDisplayName "Favourites"
```

Scripting

Executing Scripts

You can execute shells scripts, AppleScripts, JavaScript, Automator Workflows, or filter the clipboard using BBEdit Text Factories, see the [Execute Actions](#) section. You can run a script file or include the script as text directly in the application.

AppleScripts and shell scripts give you a powerful way of adding new facilities we have not specifically provided for, as well as controlling other applications. JavaScript enables deep control over a web page.

Shell scripts can execute any installed scripting language, such as perl, python, ruby or whatever. Be aware that because shell scripts are executed in a non-interactive shell, typically none of the shell configuration files (like `.login` or `.profile`) will be executed, which may change the way the shell script behaves.

The results of AppleScripts, JavaScripts and shell scripts can be displayed, or they can be typed or pasted in to the current selection, or saved into a variable or clipboard. This allows you to insert text that depends on many factors, such as date calculations, file listings, SQL queries, web pages, or anything else you can imagine.

You can also use the clipboard to pass data between actions. For example, a script can use pbpaste to read the current clipboard, and pbcopy to set the current clipboard. You can use the Delete Current Clipboard action to restore the clipboard afterwards.

Variables can also be accessed from shell scripts via the environment variables in the form \$KMVAR_Variable_Name where KMVAR_ is prefixed, and spaces are converted to underscores.

AppleScripts can also access the environment variables using the `system attribute` command, but note that `system attribute` is not safe for international characters.

AppleScript can read or write variables with:

```
tell application "Keyboard Maestro Engine"
    set kmVarRef to make variable with properties {name:"Calculation Result"}
    set oldValue to value of kmVarRef
    set value of kmVarRef to 10
end tell
```

You can delete a variable with:

```
tell application "Keyboard Maestro Engine"
    delete variable "Calculation Result"
end tell
```

JavaScript can access the variable values by using the document.kmvar dictionary, like document.kmvar.Variable_Name (spaces are converted to underscores).

AppleScripts are executed in the background via `osascript`. This means they are not allowed to do user

interaction. You can work around this by asking an application like System Events to do the user interaction for you, for example:

```
tell application "System Events"
    activate
    display dialog "Hello"
end tell
```

The `osascript` tool will execute in 64-bit mode if available, which may be a problem if you have old versions of AppleScript extensions installed. However, you can set the command line tool that is used to execute AppleScripts as described in [Other Hidden Preferences](#).

See also the [Variables](#) section.

Controlling Keyboard Maestro via Scripting

The primary scripting interface to Keyboard Maestro is the Keyboard Maestro Engine's `do script` support. You can ask Keyboard Maestro to:

- execute a macro by name
- execute a macro by unique ID
- execute an action given its XML code

Note in most cases you must ask “Keyboard Maestro Engine”, not “Keyboard Maestro”.

The easiest way is to use the name, for example:

```
tell application "Keyboard Maestro Engine"
    do script "[Name of Your Macro]"
end tell
```

The macro must be defined and enabled, and the macro group must be enabled and currently active.

If there is more than one macro with the same name, you will get an error, so you can use a `UID` instead of a name.

```
tell application "Keyboard Maestro Engine"
    do script "D0C150C7-8A0C-4837-918A-427E2BCFB6B9"
end tell
```

The `do script` will not return until the macro is finished executing.

You can determine a macro's `UID` by selecting it and choosing [Copy UID command](#) in the [Copy as sub-menu](#) in the [Edit menu](#).

An even more powerful way to script Keyboard Maestro is to execute specific actions based on their XML code. This allows you to construct any action, including changing the action on the fly, without having to create a macro first. A simple example would be:

```
tell application "Keyboard Maestro Engine"
    do script "<dict><key>MacroActionType</key><string>SwitchToLastApplication</string></dict>"
end tell
```

The easiest way to determine the appropriate XML is to create an example action in an example macro and then export the macro.

You can disable or enable a Macro or Macro Group from AppleScript with:

```
tell application "Keyboard Maestro"
    setMacroEnable "Macro/Macro Group Name or UID" with/without enable
end tell
```

This actually asks the editor to disable or enable the macro or macro group, so the change is both visible and permanent.

Alternatively you can use the Set Macro Enable action.

You can start editing a Macro or Macro Group from AppleScript with:

```
tell application "Keyboard Maestro"
    editMacro "Macro/Macro Group Name or UID"
end tell
```

You can ask Keyboard Maestro Engine to reload the macros with:

```
tell application "Keyboard Maestro Engine"
    reload
end tell
```

You can also import macros, get the selected macros, or delete a macro or macro group from AppleScript. See the [AppleScript dictionary](#) for more information.

Enhancing AppleScript

Keyboard Maestro Engine makes several of its facilities available to AppleScript.

You can ask it to play a sound with:

```
tell application "Keyboard Maestro Engine"
  play sound alias "Harddisk:System:Library:Sounds:Glass.aiff"
end tell
```

You can ask Keyboard Maestro Engine to perform a calculation for you with:

```
tell application "Keyboard Maestro Engine"
  set n to calculate "JULIANDATE()"
end tell
```

You can ask Keyboard Maestro Engine to process tokens for you with:

```
tell application "Keyboard Maestro Engine"
  set clip to process tokens "%PastClipboard%3%"
end tell
```

The keyboardmaestro URL Scheme

Another way you can control Keyboard Maestro us using the keyboardmaestro URL scheme, which supports the following formats:

- keyboardmaestro://u=support%40stairways.com/s=ABCDEFGH0123456789 – enter your username/serial number.
- keyboardmaestro://m=Activate%20Application%20Switcher – edit a specific macro or macro group.
- keyboardmaestro://q=Activate – filter macros with this keyword.
- keyboardmaestro://g=All%20Macros/q=Activate – select a macro group and filter macros with this keyword.
- keyboardmaestro://a=Execute – filter actions with this keyword.
- keyboardmaestro://c=All%20Actions/a=Execute – select action category and filter actions with this keyword.

See also the [Calculations](#) and [Text Tokens](#) sections.

Status Menu Icons

Keyboard Maestro 6 allows you to select the status menu icon.

The current lovely status menu icons were done by [Iconaholic](#), and you can also use the Classic finger tapping icons. As well as that, you can [get more](#) status menu icons from our web site and you can create and optionally contribute your own versions. You can drop a new status menu icon .zip archive on the Keyboard Maestro application dock icon to install it.

The format is fairly simple, you need a folder with the name of the icon (which must be unique relative to other contributed icons), and in that folder you have a sequence of files named StatusItem and then then the animated sequence StatusItem1, StatusItem2, etc. You can have as many as you like. If the icon is a template icon (black and clear only), then add the word “Template” to the name. And finally the extension can be either .tiff or .png. The file should be either a 16x16 icon, or ideally a retina tiff file with both a 16x16 and 32x32 at 144dpi (you can create the retina-ready tiff files using tiffutil -cathidpicheck, or you can use a tool like Opacity.app).

The two default icon folder structures look like this:

- Classic
 - StatusItem.tiff
 - StatusItem1.tiff
 - StatusItem2.tiff
 - StatusItem3.tiff
 - StatusItem4.tiff
 - StatusItem5.tiff
 - StatusItem6.tiff
- Iconaholic
 - StatusItemTemplate.tiff
 - StatusItem1Template.tiff
 - StatusItem2Template.tiff
 - StatusItem3Template.tiff
 - StatusItem4Template.tiff

Note that the “Template” appears at the end, after the animation index, and there are no spaces anywhere. You can have spaces in the folder name.

Then you archive the folder into a zip file named NameStatusMenuIcon.zip (eg ClassicStatusMenuIcon.zip) and drop it on the Keyboard Maestro application dock icon or [email it to us](#) with a cover letter indicating your permission for it to be distributed, as well as your name to be published with an optional URL if desired.

Plug In Actions

Keyboard Maestro 6 adds support for user written and contributable plug in actions. You can [get more](#) plug in actions from our web site and you can create and optionally contribute your own. You can drop a new plug in action .zip archive on the Keyboard Maestro application dock icon to install it (note that to update a plug in action you must manually remove it from the `~/Library/Application Support/Keyboard Maestro/Keyboard Maestro Actions` folder before re-installing it).

A Third Party Plug In Action consists of a folder with a name (which should generally closely match the action name), and must be made up of only ASCII alphanumerics, underscores and spaces. The folder name must be unique among all plug in actions. The folder name is stored in the Keyboard Maestro Macros.plist to reference the plugin action.

The folder contains a set of files, including:

- Keyboard Maestro Action.plist – an XML file describing the action.
- A script file whose name must be made up of only ASCII alphanumerics or underscores, plus an ASCII alphanumeric extension. It may be a shell script or an AppleScript. If it is a shell script, it will be made executable automatically.
- An optional 64x64 png icon.

The format of the Keyboard Maestro Action.plist is a Cocoa property list containing a dictionary with the following keys and values:

Name

the name of the action (which appears in the Category/Actions list)

Script

the name of the script, made up of only ASCII alphanumerics or underscores, plus an ASCII alphanumeric extension.

Icon [optional]

the name of the icon png file, made up of only ASCII alphanumerics or underscores plus .png.

Title [optional]

the title displayed on the action, which can include %Param%XYZ% tokens. It should usually not include other tokens. If it is missing, the Name will be used.

Timeout [optional number]

the default timeout in seconds. Set it to 0 if the action needs no timeout. The default is 99 hours.

Author [optional]

the author of this action.

URL [optional]

a URL for the author or this action.

Help [optional]

a short (Tool Tip) explanation of this action.

Results [optional]

what to do with the output of the script if any. Possible Values: None, Window, Briefly, Typing, Pasting, Variable, Clipboard – multiple values can be used, seperated by a bar (|), the first specified value is the default.

Parameters [optional]

an array of parameters to the script, each entry is a dictionary as described below.

Each parameter in the Parameters array is a dictionary with the following keys:

Label

the name of the parameter. The same rules as Keyboard Maestro Variable Names apply. The label is displayed to the user and used to pass the parameter to the script. Obviously, the label must be unique amongst all parameters.

Type

the type of the parameter. Possible Values: String (single line), TokenString, Calculation, Text (multi-line), TokenText, Checkbox (0 or 1), PopupMenu or Hidden. The Type specifies how the value is displayed to the user and what processing is applied before it is passed to the script. Hidden types are text token processed, but are not displayed in the editor.

Default [optional]

the default value when the action is created.

Menu [required if Type is PopupMenu]

the values of the popup menu, separated by |.

Warning: Keys are case sensitive.

Parameters are passed to the script via environment variables with names starting with KMPARAM_ similar to how variables are passed to shell scripts with the Execute Script action. So a parameter named "My Text"

would be in an environment variable `KMPARAM_My_Text`. You can access the environment variables from AppleScript with `system attribute "KMPARAM_My_Text"`. Note that AppleScript's `system attribute` is not safe for international characters, although can use code like:

```
set myText to do shell script "echo $KMPARAM_My_Text"
```

In normal use, once a plug in action is read, it will stay in memory and changes will not be noticed (although the script will be executed each time, so changes to that will be noticed). To cause the editor and/or engine to notice changes to the plug while in development, use AppleScript to reload the macros:

```
tell application "Keyboard Maestro" to reload
tell application "Keyboard Maestro Engine" to reload
```

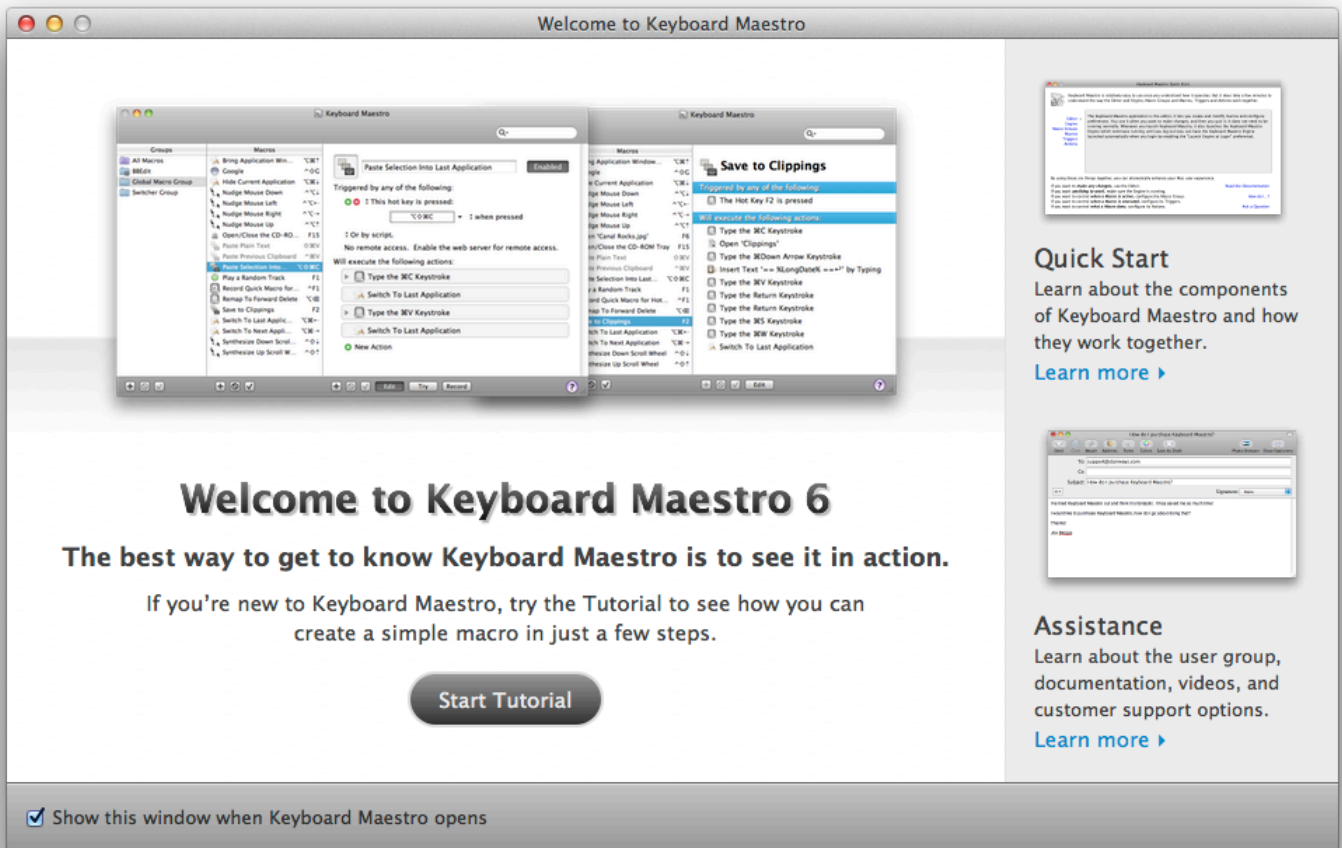
Keyboard Maestro Plug In Actions go in the `~/Library/Application Support/Keyboard Maestro/Keyboard Maestro Actions` folder.

Windows

- [Welcome Window](#)
- [Macros Window](#)
- [Tutorial](#)
- [Macro Group Editor](#)
- [Macro Editor Window](#)
- [Macro Library Window](#)
- [Icon Chooser Window](#)
- [Recording Window](#)
- [Macro Debugger](#)
- [Conflict Palette](#)
- [Trigger Macro by Name](#)
- [Application Launcher Window](#)
- [Application Switcher Window](#)
- [Window Switcher Window](#)
- [Clipboard Switcher Window](#)
- [Clipboard History Switcher Window](#)
- [Preferences Window](#)
- [Preferences General Pane](#)
- [Preferences Web Server Pane](#)
- [Preferences Variables Pane](#)
- [Preferences Clipboards Pane](#)
- [Preferences Exclude Pane](#)
- [About Window Pane](#)

Welcome Window

This window welcomes new users and gives you some options for learning about Keyboard Maestro.

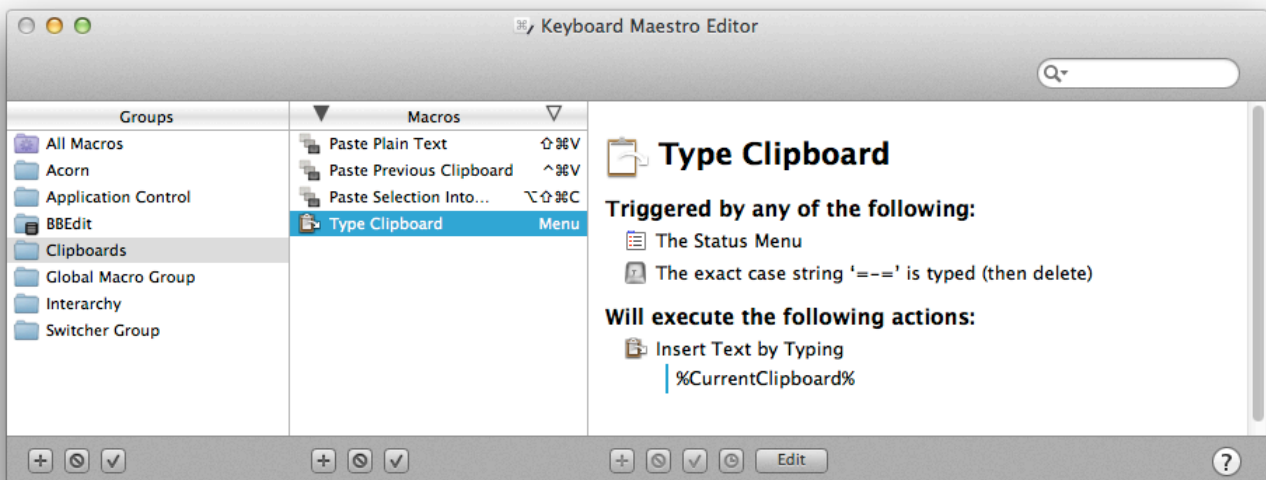


If you are new to Keyboard Maestro, start the tutorial and Keyboard Maestro will walk you through creating a simple macro.

Macros Window


This window lets you manipulate Macros and Macro Groups, creating new ones, deleting old ones, enabling and disabling them, editing them and so on.

You get this window pane by launching Keyboard Maestro.



The window contains a list of Macro Groups and their associated Macros.


You can create a new Macro Group by clicking the  button below the Groups list.


You can create a new Macro by selecting a macro group and then clicking the  button below the Macros list.

You can see the selected Macro Group or Macro in the right hand column, and edit it by clicking the Edit button.

You can select the All Macros meta-Group to show all Macros, and you can use the search field to filter down the list of macros.

You can rename a Macro Group or Macro by double clicking it and changing the title.

You can delete a Macro Group by selecting it and clicking the  button below the Groups list.

You can delete a Macro by selecting it and clicking the  button below the Macros list.

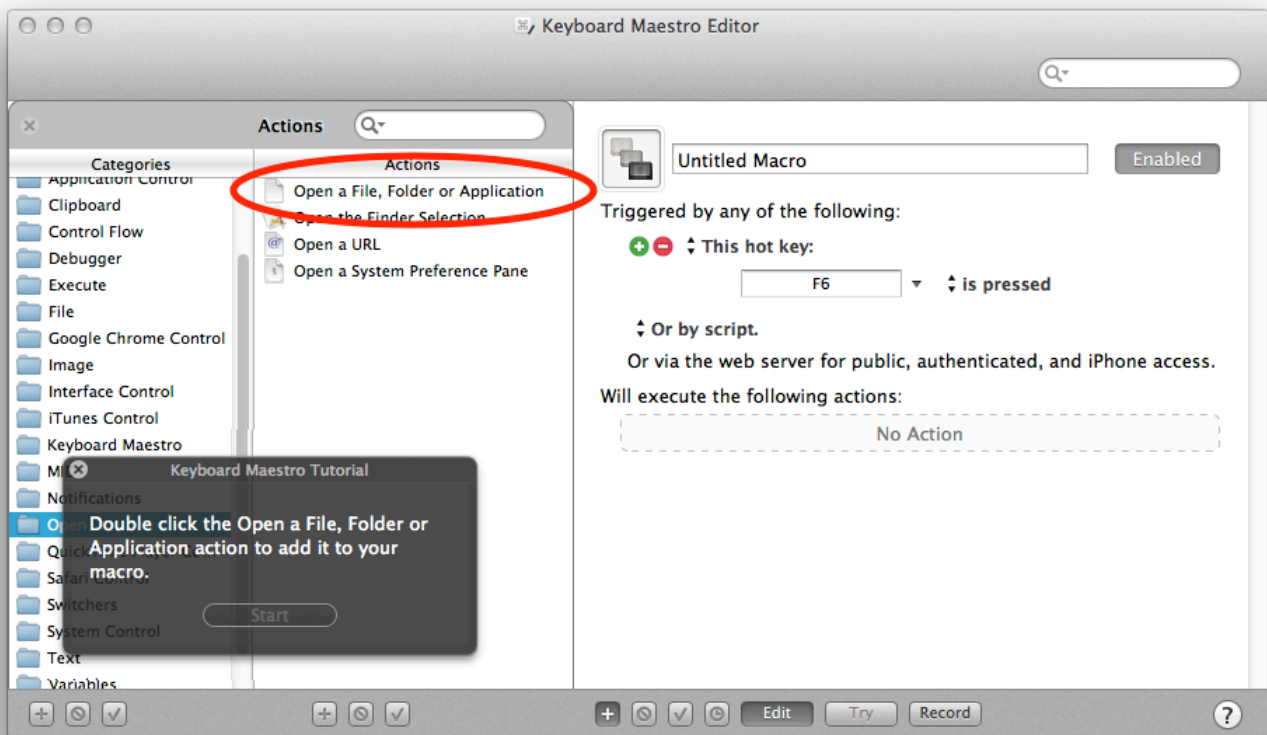
Similarly, you can enable or disable Macro Groups or Macros by clicking their respective  button.

You cannot delete, rename or modify the Global Macro Group.

See also the [Macro Groups](#), [Macros](#) and [Macro Editor Window](#) sections.

Tutorial

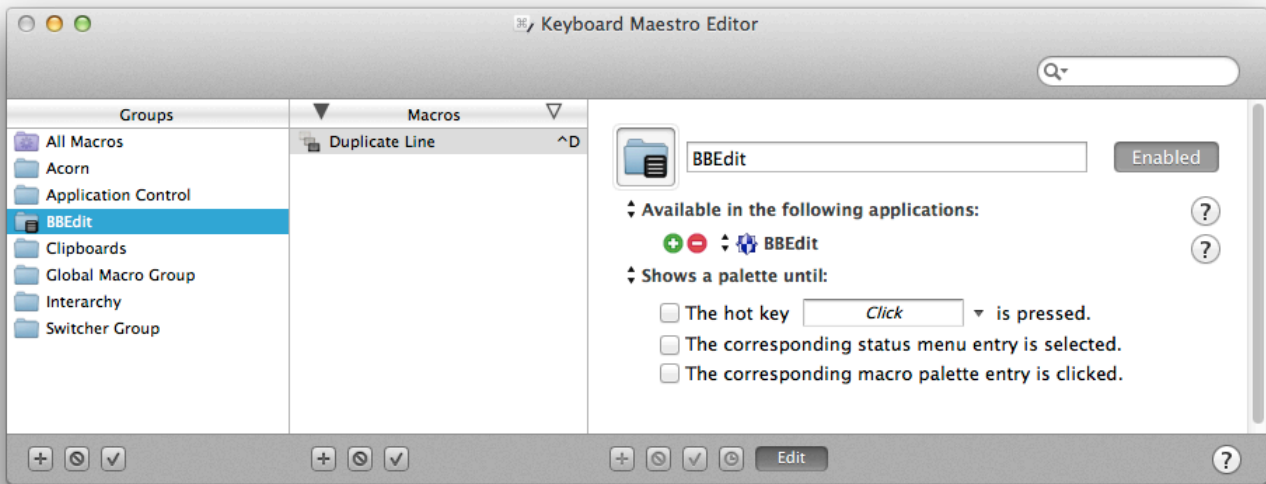
If you are new to Keyboard Maestro, start the tutorial by clicking the [Start Tutorial](#) button in the [Welcome window](#) or by choosing [Tutorial](#) from the [Help menu](#) and Keyboard Maestro will walk you through creating a simple macro.



Follow the instructions. Keyboard Maestro will highlight the location of the various buttons to help you quickly create a macro. You can even use the tutorial as a wizard to create a hot key triggered macro to perform any of Keyboard Maestro's many actions.

Macro Group Editor

To edit a Macro Group, select it and click the Edit button. Its details will be shown in the right hand column. You can edit its name, control which applications it is available in, and how it will be activated.



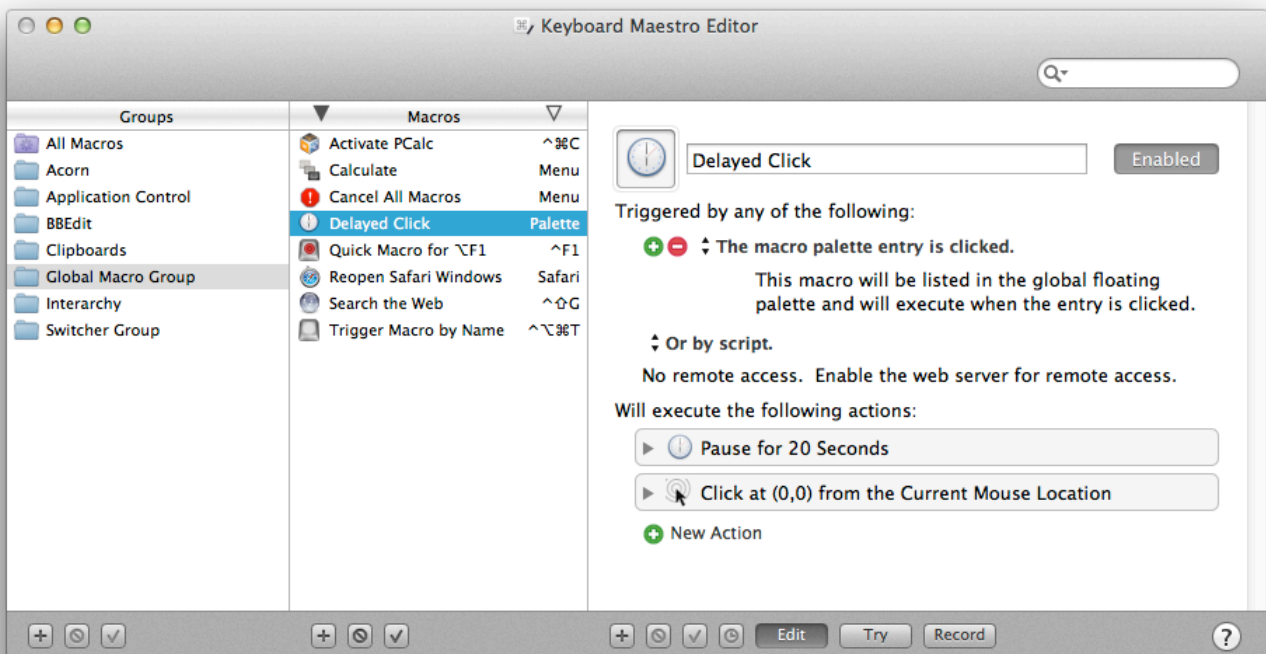
Typically a group's macros would be available everywhere (available in all applications), or it might be specific to a particular application (available in the following applications) in which case you might name the Macro Group after the application.

You can also configure the macro group to be activated only after a Hot Key press (either for a single use or toggled on and off), and whether to display the macros in a floating palette.

See also the [Macros](#) section.


Macro Editor Window



To edit a Macro, select it and click the Edit button. Its details will be shown in the right hand column. You can edit its name, add or remove triggers, and configure its action list.




To add a trigger, click the green button and select the type of trigger. To remove a trigger, click the red button.

To see how to execute this macro via a script, select from the "Or by script" menu. How you can execute the macro remotely is also displayed.

To add an action, click the New Action button, or equivalently the  button below the detail view. This will show the lists of possible actions. Double click one or more of them to add actions to the action list for this macro.

You can also Copy and Paste actions, as well as drag them around to rearrange them. Use the  button and  button to delete or enable/disable the selected actions.

You can try the selected actions by clicking the  button.

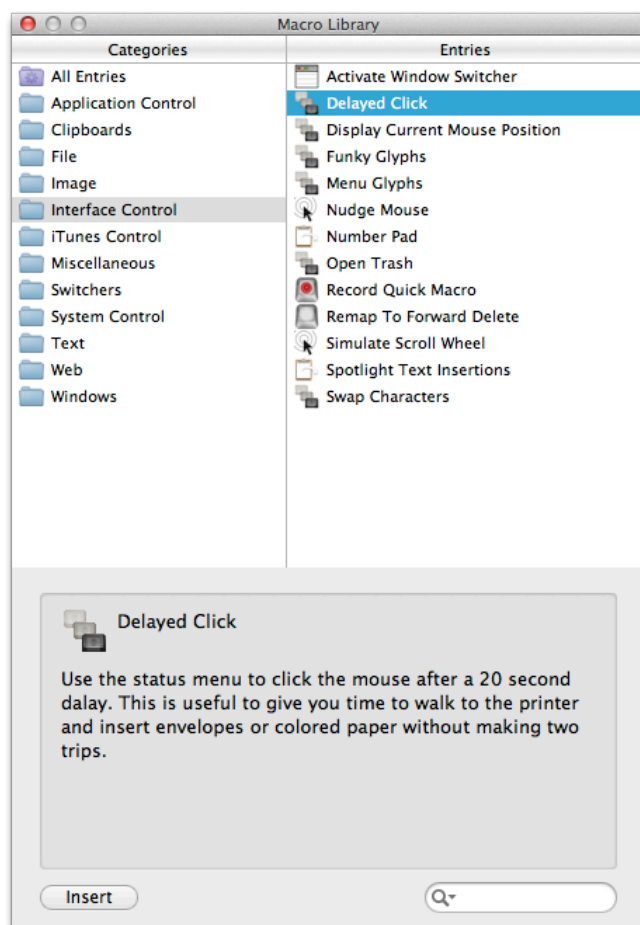
You can click the  button to record your actions.


To learn more about creating or editing Macros, see the [Macros](#) section.

Macro Library Window

This window contains example and template macros you can add to your macro collection. You can use the macros as is, or edit them to customize them for your particular needs.

You get this window by choosing [Macro Library](#) from the [Window menu](#).



Each entry represents one or more macros, usually in a single macro group, but occasionally in more than one macro group. You can learn about them by selecting them, and then you can insert them into your macros by clicking the  button or by dragging them to a particular macro group (dragging is not available if the library entry represents more than one macro group as you can't drag to two macro groups).

You can add macros to your library by choosing [Export as Macro Library](#) from the [File menu](#) and selecting the Add to My Macro Library checkbox. You can get Macro Library entries from us or from other Keyboard Maestro users and add them to you library by double clicking them or by choosing [Import to Macro Library](#) from the [File menu](#).

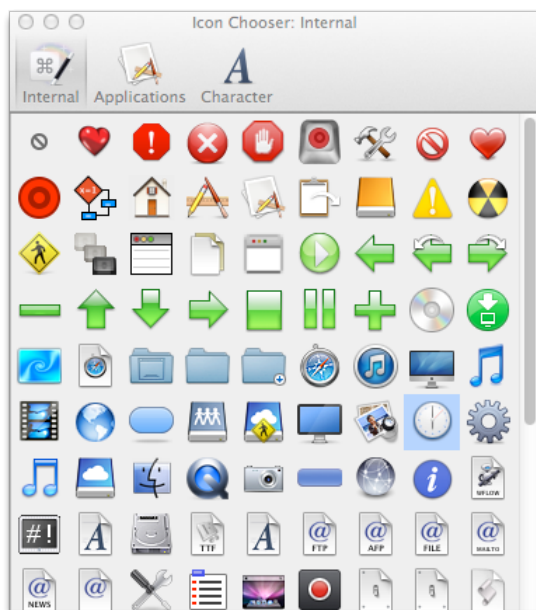
Remember to use caution when installing a macro or macro library from anyone – macros can potentially do a lot of damage and compromise the security of your Mac, so only install macros from trusted sources.

Icon Chooser Window

This window contains icons you can use to customize your macro and macro groups. You can click on an icon well in the macro or macro group editor and then show this window and select or create an icon for it.

You get this window by choosing Icon Chooser from the Window menu.

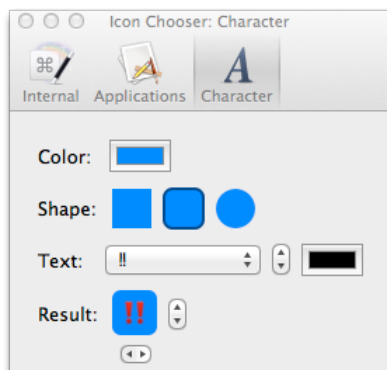
There are three types of icons available. Internal Keyboard Maestro icons:



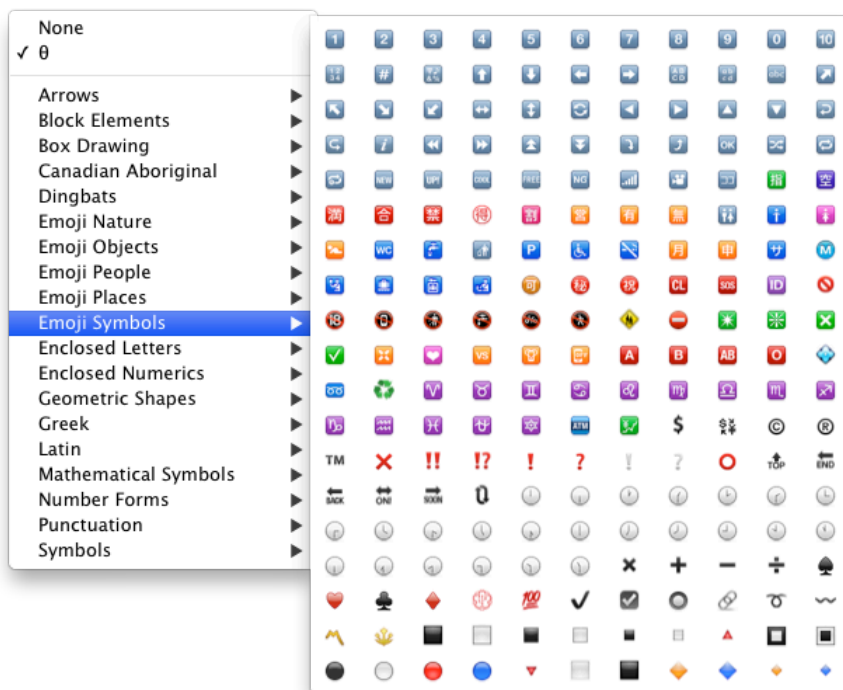
application icons available on your Mac:



and icons you create by choosing a shape, color and optional character:



When creating icons, you can reference lots of characters:

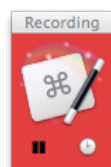


Alternatively, you can copy an image from anywhere and paste it in to the icon well (although Keyboard Maestro stores a small reference to the Icon Chooser icons, so that is much more efficient than storing the image in your macros as copy & paste will).

Recording Window

This window shows you when Keyboard Maestro is recording your actions.

You get this window by clicking the **Record** button in the Macro Editor window or by triggering a Record Quick Macro action.

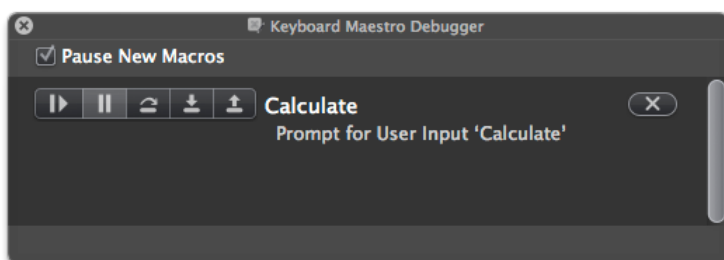


Clicking on this window will stop all recording, or you can pause recording, or add a 1 second Pause action to the current recording.

To learn more about recording, see the Recording section.

Macro Debugger

You get this window by choosing Start Debugging from the Status Menu menu by triggering a Debugger Start action.



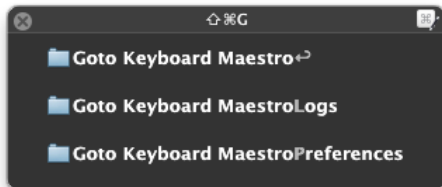
To learn more about debugging, see the Macro Debugger section.

Conflict Palette

You get a conflict palette when a Hot Key would trigger two or more actions.



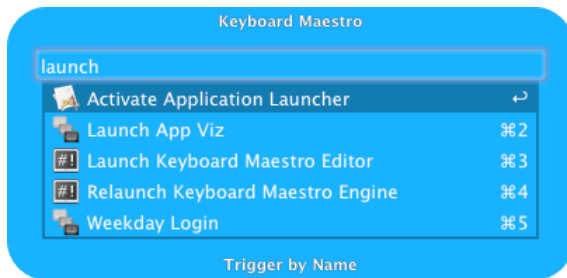
Note how the distinguishing characters are highlighted – pressing those keys will select the desired trigger, or filter the list so that only those macros remain:



this continues until only one macro remains which is immediately executed.

Trigger Macro by Name

You get this window by executing the Trigger Macro by Name action.



Type a string to filter the macros.

Note that the filtering is not just by macro name.

Application Launcher Window

This window lets you launch applications.

You get this window by triggering the Activate Application Launcher macro.

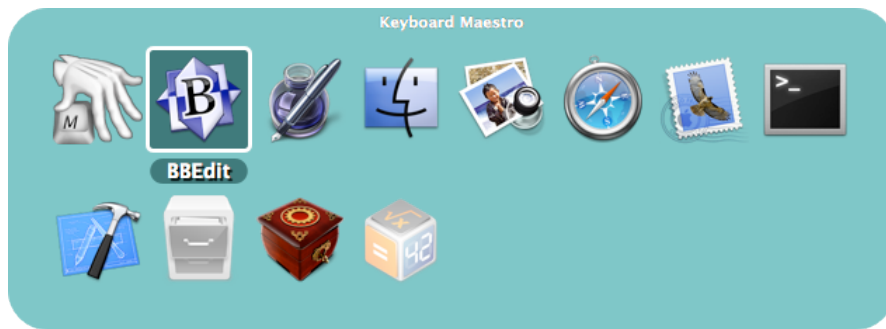


To learn more about the Application Launcher, see the [Application Launcher](#) section.

Application Switcher Window

This window lets you switch between active applications, as well as quit or hide applications or quickly launch frequently used applications.

You get this window by triggering the Activate Application Switcher macro.

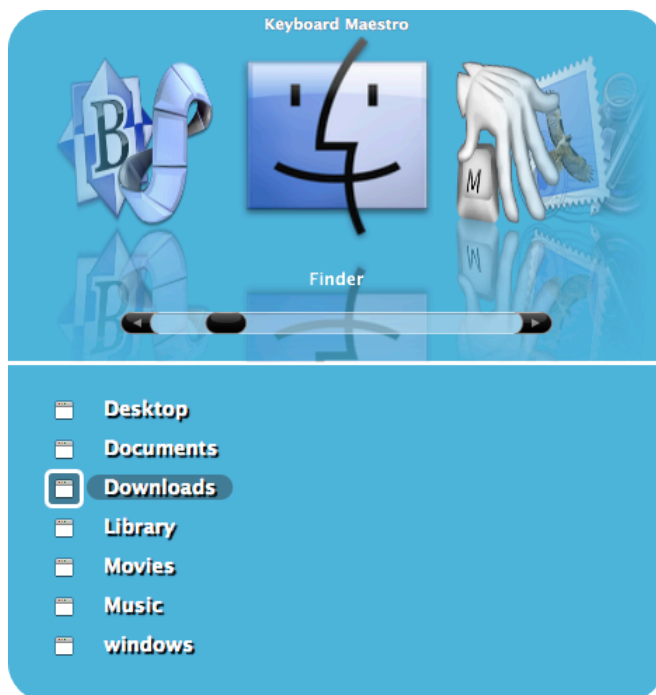


To learn more about the Application Switcher, see the [Application Switcher](#) section.

Window Switcher Window

This window lets you switch between windows in the current application, as well as close or minimize windows.

You get this window by triggering the Activate Window Switcher macro.

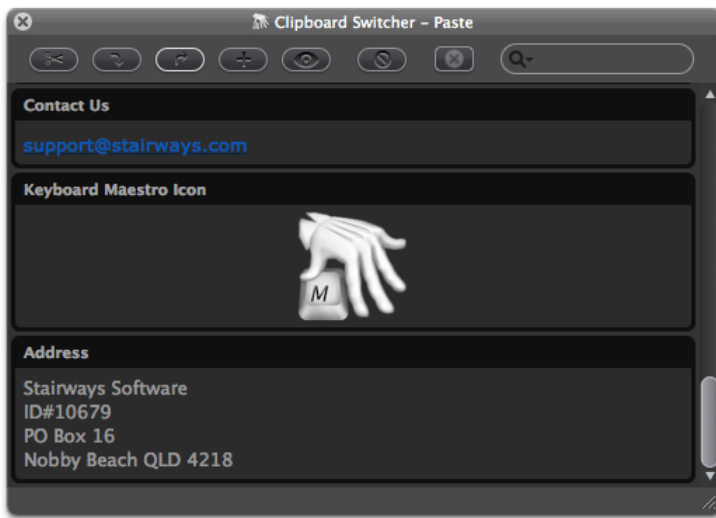


To learn more about the Window Switcher, see the [Window Switcher](#) section.

Clipboard Switcher Window

This window lets you select between named clipboards to Cut, Copy or Paste to/from.

You get this window by triggering one of the Activate Clipboard Switcher macros.



Select something and select a named clipboard and click the **Cut** button or **Copy** button to cut/copy to a named clipboard. Select a named clipboard and click the **Paste** button to paste a named clipboard. Click the **+** button to create a new named clipboard. Select a named clipboard and click the **Quick Look** button to view it. Select a named clipboard and click the **☒** button to delete it.

Click the **x** button to toggle whether the window should close after an action.

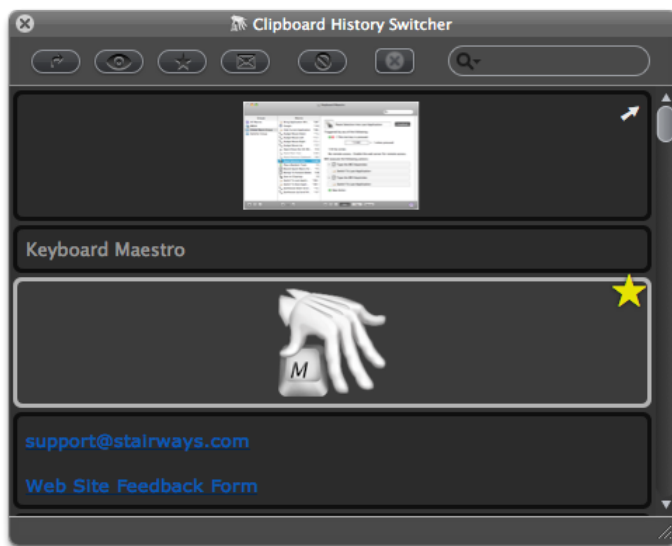
Use the search field to filter the named clipboards.

To learn more about the Clipboard Switcher, see the [Clipboard Switcher](#) section.

Clipboard History Switcher Window

This window lets you paste from your clipboard history of items that you have previously cut or copied.


You get this window by triggering the Clipboard History Switcher macro.

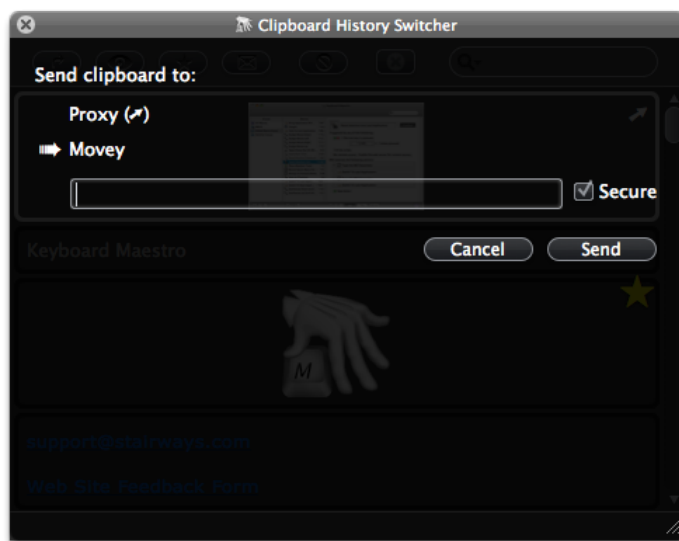







Cut or copy something and it will appear in this list. Select an item and click the **Paste** button to paste a named clipboard. Select an item and click the **★** button (or press Command-L) to mark it as a favorite, or click the **✉** button (or press Command-S) to send it to another Mac. Select an item and click the **☒** button to delete it (this is useful if you want to delete a password or other sensitive information).


To view an item more fully, select it and click the **Quick Look** button (or press Space) to display the [Quick Look window](#).



To send a clipboard to another Mac running Keyboard Maestro, click the  button to display the sending window.



Select the desired local destination, or type a host:port destination to send it to a remote Mac, and click the send button. Keyboard Maestro will keep trying to send to the Mac even if it can't connect right now, so as long as both Macs are connected to the Internet eventually, the clipboard should get through. While a clipboard is being sent, it is marked with a . If a clipboard is marked to be sent in the future, it is marked with a . Once it has been successfully sent, it is marked with a . If it fails to be sent, and Keyboard Maestro has given up, it is marked with a . Clipboards that have been received are marked with a .

Click the  button to toggle whether the window should close after an action.

Use the search field to filter the named clipboards.

To learn more about the Clipboard History Switcher, see the [Clipboard History Switcher](#) section.

Preferences Window

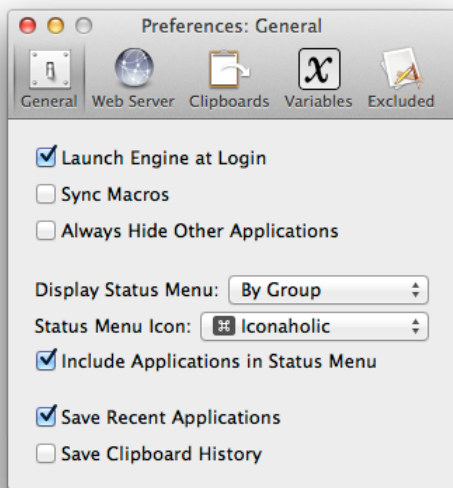
This window lets you configure Keyboard Maestro.

You get this window by launching Keyboard Maestro and choosing [Preferences](#) from the [Keyboard Maestro menu](#).

To learn more about the Preferences, see the [Preferences](#) section.

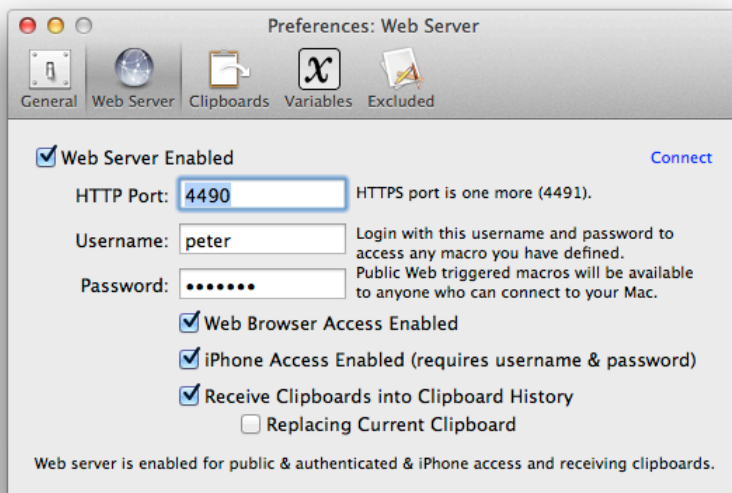
Preferences General Pane

This window pane lets you configure general preferences.



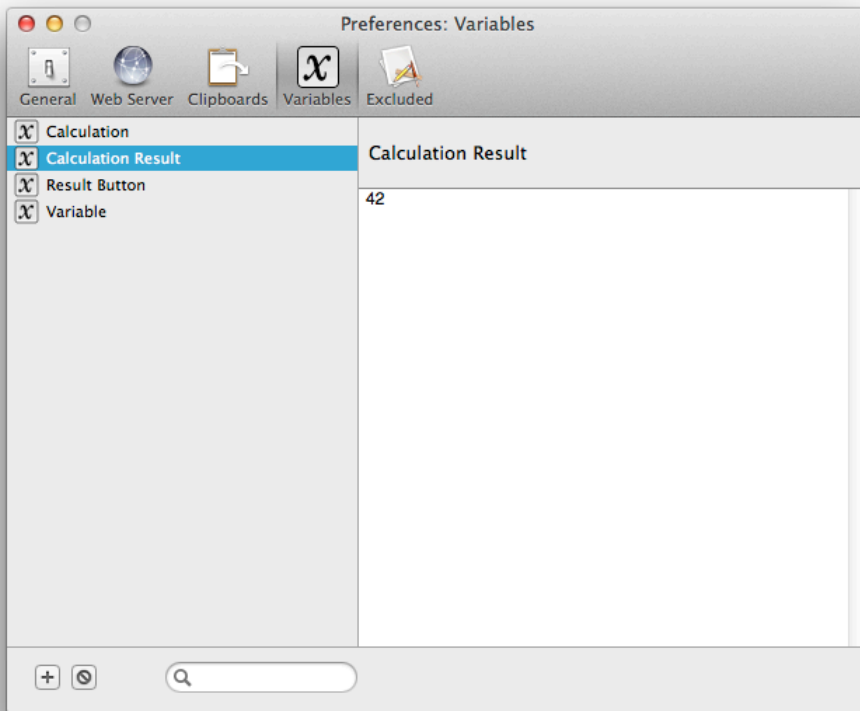
Preferences Web Server Pane

This window pane lets you configure the built-in web server which enables remote execution of macros.



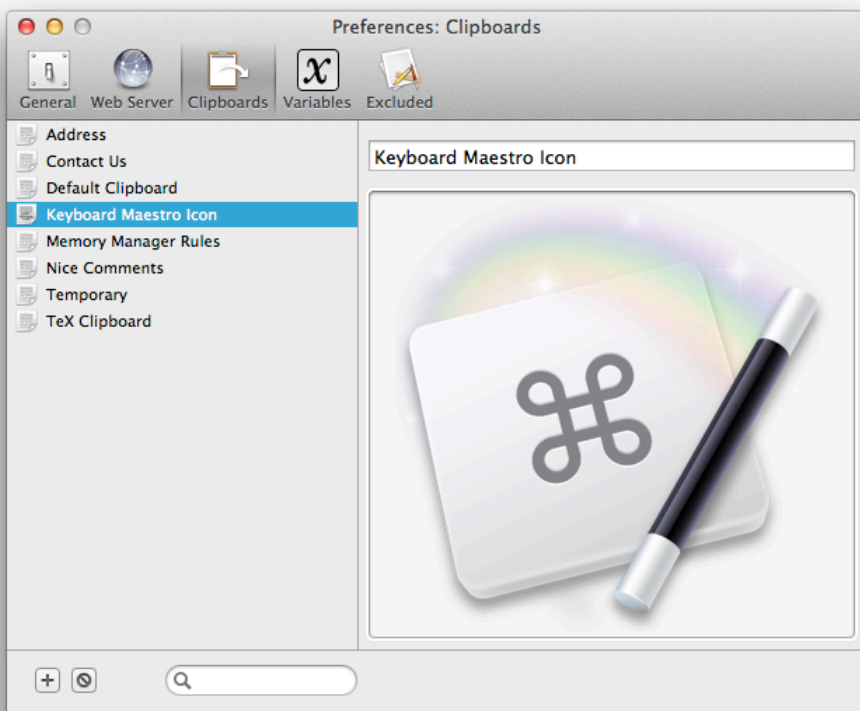
Preferences Variables Pane

This window pane lets you add and remove variables, and see and change their values.



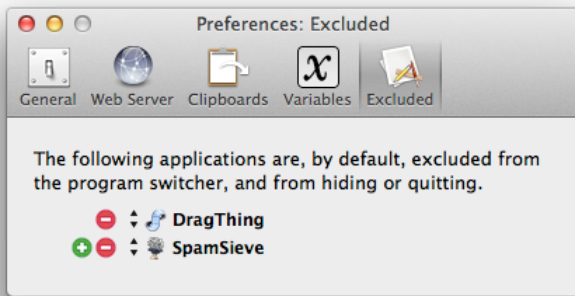
Preferences Clipboards Pane

This window pane lets you add, remove and rename Named Clipboards and see and change their values.



Preferences Exclude Pane

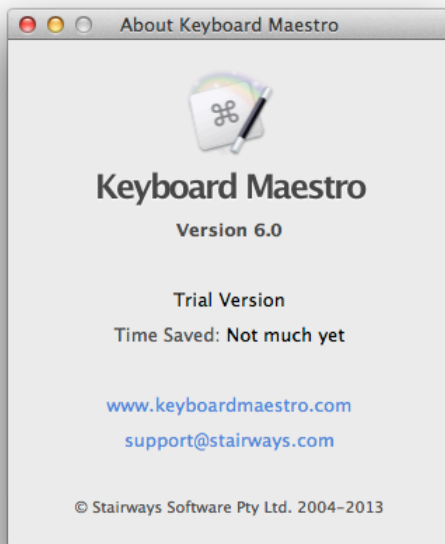
This window pane lets you add and remove applications from the global Excluded Applications list.



About Window Pane

This window shows you the version of this copy of Keyboard Maestro, to whom it is registered, and allows you to visit the web site.

You get this window by launching Keyboard Maestro and choosing About Keyboard Maestro from the Keyboard Maestro menu.



Menus

Keyboard Maestro

- About Keyboard Maestro
- Purchase Keyboard Maestro
- Register Keyboard Maestro
- Check For Update
- Preferences
- Services
- Hide Keyboard Maestro
- Hide Others
- Show All
- Quit Keyboard Maestro

File

- New Macro Group
- New Macro
- Close
- Export Macros
- Export as Macro Library
- Import Macros

- [Import to Macro Library](#)
- [Revert Macros](#)
- [Start Syncing Macros](#)
- [Reveal Macro Sync File](#)
- [Stop Syncing Macros](#)
- [Launch Engine](#)
- [Quit Engine](#)

Edit

- [Undo](#)
- [Redo](#)
- [Cut](#)
- [Copy](#)
- [Copy as](#)
- [Paste](#)
- [Delete](#)
- [Select All](#)
- [Duplicate](#)
- [Insert Token](#)
- [Insert Function](#)

View

- [Sort by Macro Name](#)
- [Sort by Macro Trigger](#)
- [Sort by Macro Modification](#)
- [Sort by Macro Execution](#)
- [Expand Action](#)
- [Collapse Action](#)
- [Start Editing Macros](#)
- [Stop Editing Macros](#)
- [Toggle Enable](#)
- [Show Actions](#)
- [Set Action Timeout](#)
- [Try](#)
- [Record](#)

Window

- [Minimize](#)
- [Zoom](#)
- [Keyboard Maestro Editor](#)
- [Macro Library](#)
- [Icon Chooser](#)
- [Bring All to Front](#)

Help

- [Keyboard Maestro Documentation](#)
- [Keyboard Maestro Quick Start](#)
- [Welcome to Keyboard Maestro](#)
- [Tutorial](#)
- [Videos](#)
- [Open Logs Folder](#)
- [Open Preferences Folder](#)
- [Third Party Licenses](#)
- [Online Documentation](#)
- [Keyboard Maestro Web Site](#)
- [Stairways Software Web Site](#)
- [Report Bugs or Feature Requests](#)
- [Service and Support](#)

Status Menu

- [Launch Keyboard Maestro Editor](#)
- [Cancel](#)
- [Start Debugging](#)
- [Quit Keyboard Maestro Engine](#)

Keyboard Maestro

The [Keyboard Maestro menu](#) contains menu items relating to the Keyboard Maestro application as a whole.

Keyboard Maestro » About Keyboard Maestro

The [About Keyboard Maestro](#) command in the [Keyboard Maestro menu](#) displays the [About Keyboard Maestro](#) window.

Keyboard Maestro » Purchase Keyboard Maestro

The [Purchase Keyboard Maestro command](#) in the [Keyboard Maestro menu](#) lets you purchase Keyboard Maestro online.

Keyboard Maestro » Purchase Keyboard Maestro Upgrade

The [Purchase Keyboard Maestro Upgrade command](#) in the [Keyboard Maestro menu](#) lets you purchase an upgrade to Keyboard Maestro online.

Keyboard Maestro » Register Keyboard Maestro

The [Register Keyboard Maestro command](#) in the [Keyboard Maestro menu](#) displays the serial number entry window allowing you to enter your username (email address) and serial number. Make sure you enter them exactly as sent to you.

Keyboard Maestro » Check For Update

The [Check For Update command](#) in the [Keyboard Maestro menu](#) checks to see if there are any updates to Keyboard Maestro and offers to download and install them if there are.

Keyboard Maestro » Preferences

The [Preferences command](#) in the [Keyboard Maestro menu](#) displays the [Preferences window](#).

Keyboard Maestro » Services

The [Services command](#) in the [Keyboard Maestro menu](#) is used to perform [Mac OS X Services](#) which are shared functions available across multiple applications. You can learn more about [Mac OS X Services](#) from your [Mac OS X documentation](#), and you can install new services which will work with Keyboard Maestro. Keyboard Maestro includes full support for Services, so relevant Services on your system are available in Keyboard Maestro.

Keyboard Maestro » Hide Keyboard Maestro

The [Hide Keyboard Maestro command](#) in the [Keyboard Maestro menu](#) will hide the Keyboard Maestro application and all its windows. Click on Keyboard Maestro's Dock icon or choose Show All to show Keyboard Maestro again.

Keyboard Maestro » Hide Others

The [Hide Others command](#) in the [Keyboard Maestro menu](#) will hide all other applications. Choose Show All to show them again.

Keyboard Maestro » Show All

The [Show All command](#) in the [Keyboard Maestro menu](#) will show all hidden applications.

Keyboard Maestro » Quit Keyboard Maestro

The [Quit Keyboard Maestro command](#) in the [Keyboard Maestro menu](#) will Quit Keyboard Maestro. the [Keyboard Maestro Engine](#) will remain running and all enabled Keyboard Maestro features will continue to operate (unless you have specifically quit the [Keyboard Maestro Engine](#)).

File

The [File menu](#) is where you import or export Macros or launch or quit the [Keyboard Maestro Engine](#).

File » New Macro Group

The [New Macro Group command](#) in the [File menu](#) creates and starts editing a new macro group.

File » New Macro

The [New Macro command](#) in the [File menu](#) creates and starts editing a new macro.

File » Close

The [Close command](#) in the [File menu](#) closes the front window.

File » Export Macros

The [Export Macros command](#) in the [File menu](#) exports the selected macros to a file that you can import on another Mac. This is one way to transfer macros from one Mac to another.

File » Export as Macro Library

The [Export as Macro Library command](#) in the [File menu](#) exports the selected macros to a library file that you can share with others. If you create any interesting macros please consider sending them to us and we will make them available on our web site or in a future version of Keyboard Maestro.

See also the [Macro Library](#) section.

File » Import Macros

The [Import Macros command](#) in the [File menu](#) lets you select a saved macro file and imports the macros it contains.

File » Import to Macro Library

The [Import to Macro Library command](#) in the [File menu](#) lets you import a shared macro library file into your macro library. Macros in your library are not active, but can be added into one or more macro groups to become active.

See also the [Macro Library](#) section.

File » Revert Macros

The [Revert Macros command](#) in the [File menu](#) lets you revert to a previous version of your macros. If you find you have really messed up your macros, you can revert to how they were when you first launched Keyboard Maestro, or how they were yesterday or even several days ago.

File » Start Syncing Macros

The [Start Syncing Macros command](#) in the [File menu](#) lets you start syncing your macros with another Mac.

See also the [Macro Syncing](#) section.

File » Reveal Macro Sync File

The [Reveal Macro Sync File command](#) in the [File menu](#) lets you see where your macro sync file is.

See also the [Macro Syncing](#) section.

File » Stop Syncing Macros

The [Stop Syncing Macros command](#) in the [File menu](#) lets you stop syncing your macros.

See also the [Macro Syncing](#) section.

File » Launch Engine

The [Launch Engine command](#) in the [File menu](#) lets you start the [Keyboard Maestro Engine](#) manually. The [Keyboard Maestro Engine](#) performs all the [Macro](#), [Application Switcher](#), [Window Switcher](#) and [Clipboard Switcher](#) functions even while Keyboard Maestro itself is not running. It is launched automatically as a Startup Item when you login (assuming you have not disabled that in the [Preferences window](#)) or any time you launch Keyboard Maestro. If it is not running for any reason you can start it manually with this command. This menu item only exists while the [Keyboard Maestro Engine](#) is not running.

File » Quit Engine

The [Quit Engine command](#) in the [File menu](#) lets you quit the [Keyboard Maestro Engine](#). The [Keyboard Maestro Engine](#) performs all the [Macro](#), [Application Switcher](#), [Window Switcher](#) and [Clipboard Switcher](#) functions even while Keyboard Maestro itself is not running. It is launched automatically as a Startup Item when you login (assuming you have enabled that in the [Preferences window](#)). If you quit the [Keyboard Maestro Engine](#) these functions will no longer operate. This menu item only exists while the [Keyboard Maestro Engine](#) is running.

Edit

The [Edit menu](#) contains menu items relating to text and selections.

Edit » Undo

The [Undo command](#) in the [Edit menu](#) undoes the previous command.

Edit » Redo

The [Redo command](#) in the [Edit menu](#) redoes the previous undone command.

Edit » Cut

The [Cut command](#) in the [Edit menu](#) copies the current selection to the system clipboard and then deletes the selection.

Edit » Copy

The [Copy command](#) in the [Edit menu](#) copies the current selection to the system clipboard.

Edit » Copy as

The [Copy as sub-menu](#) in the [Edit menu](#) menu allows you to copy the selection in a variety of formats.

Edit » Copy as » Copy as Text

The [Copy as Text command](#) in the [Copy as sub-menu](#) in the [Edit menu](#) menu allows you to copy the selected macro or actions as styled text.

Edit » Copy as » Copy as Image

The [Copy as Image command](#) in the [Copy as sub-menu](#) in the [Edit menu](#) menu allows you to copy the selected macro or actions as an image.

Edit » Copy as » Copy UID

The [Copy UID command](#) in the [Copy as sub-menu](#) in the [Edit menu](#) menu allows you to copy the selected macro or macro group's [UID](#).

Edit » Paste

The [Paste command](#) in the [Edit menu](#) pastes the current system clipboard into the current selection.

Edit » Delete

The [Delete command](#) in the [Edit menu](#) deletes the current selection.

Edit » Select All

The [Select All command](#) in the [Edit menu](#) selects all text or items.

Edit » Duplicate

The [Duplicate command](#) in the [Edit menu](#) duplicates the selected items.

Edit » Insert Token

The [Insert Token command](#) in the [Edit menu](#) lets you insert any of the many [Text Tokens](#) available in Keyboard Maestro. The token will be inserted, together with an example format of any parameters it takes.

Most text fields in Keyboard Maestro can process Text Tokens.

Edit » Insert Function

The [Insert Function command](#) in the [Edit menu](#) lets you insert any of the many [functions](#) available in Keyboard Maestro. The functions will be inserted, together with an example of any parameters it takes.

Most numeric fields in Keyboard Maestro can process an expression containing these functions.

View

The [View menu](#) contains menu items relating to display and actions.

View » Sort by Macro Name

The [Sort by Macro Name command](#) in the [View menu](#) sorts the macros in the main window by name.

View » Sort by Macro Trigger

The [Sort by Macro Trigger command](#) in the [View menu](#) sorts the macros in the main window by trigger. This is useful to see what hot keys are available, or to group all Typed String triggers together.

View » Sort by Macro Modification

The [Sort by Macro Modification command](#) in the [View menu](#) sorts the macros in the main window by their modification date (most recently modified at the top). This is useful to see what macros you have recently modified.

View » Sort by Macro Execution

The [Sort by Macro Execution command](#) in the [View menu](#) sorts the macros in the main window by their last execution date (most recently executed at the top). This is useful to see what macros you have recently executed, especially if you think one might have been executed inadvertently.

View » Expand Action

The [Expand Action command](#) in the [View menu](#) expands (discloses) the selected action(s). Hold the option key down to expand all actions in a given sublist.

View » Collapse Action

The [Collapse Action command](#) in the [View menu](#) collapses the selected action(s). Hold the option key down

to collapse all actions in a given sublist.

View » Start Editing Macros

The [Start Editing Macros command](#) in the [View menu](#) turns on edit mode. Edit mode allows you to modify macros and macro groups. You may prefer to leave it on permanently, or you may like to turn it off when you are not making changes to give a more concise and visually appealing view of the macros and macro groups.

View » Stop Editing Macros

The [Stop Editing Macros command](#) in the [View menu](#) turns off edit mode.

View » Toggle Enable

The [Toggle Enable command](#) in the [View menu](#) toggles the enable of the selected items.

View » Show Actions

The [Show Actions command](#) in the [View menu](#) shows the action list. It is available while editing a macro. Double click or drag actions from the action list to add them to your macro. This menu toggles to Hide Actions when the action list is already showing.

View » Set Action Timeout

The [Set Action Timeout command](#) in the [View menu](#) sets the timeout time and behaviour for the selected actions. You can configure how long an action is allowed to execute for before it is aborted, and whether the macro should continue or be canceled if the action times out.

View » Try

The [Try command](#) in the [View menu](#) tries the selected actions. It is available while editing a macro.

View » Record

The [Record command](#) in the [View menu](#) toggles recording on and off. It is available while editing a macro.

Window

The [Window menu](#) contains menu items relating to windows.

Window » Minimize

The [Minimize command](#) in the [Window menu](#) minimizes the front window.

Window » Zoom

The [Zoom command](#) in the [Window menu](#) zooms the front window.

Window » Keyboard Maestro Editor

The [Keyboard Maestro Editor command](#) in the [Window menu](#) brings the main Keyboard Maestro macro editing window to the front.

Window » Macro Library

The [Macro Library command](#) in the [Window menu](#) shows or hides the macro library.

See also the [Macro Library](#) section.

Window » Icon Chooser

The [Icon Chooser command](#) in the [Window menu](#) shows or hides the Icon Chooser.

See also the [Icon Chooser](#) section.

Window » Bring All to Front

The [Bring All to Front command](#) in the [Window menu](#) brings all Keyboard Maestro windows to the front.

Help

The [Help menu](#) contains menu items relating to Help.

Help » Keyboard Maestro Documentation

The [Keyboard Maestro Documentation command](#) in the [Help menu](#) displays the Keyboard Maestro documentation.

Help » Keyboard Maestro Quick Start

The [Keyboard Maestro Quick Start command](#) in the [Help menu](#) displays the Keyboard Maestro quick start help, which quickly gets you up to speed in using Keyboard Maestro.

Help » Welcome to Keyboard Maestro

The [Welcome to Keyboard Maestro command](#) in the [Help menu](#) displays the Welcome message, giving you a quick overview of what support resources are available for helping you get started using Keyboard Maestro.

Help » Tutorial

The [Tutorial command](#) in the [Help menu](#) starts the in-application tutorial. The tutorial will walk you through creating a simple macro. By varying the actions slightly, you can create a variety of macros that are triggered by hot keys and that open various documents.

Help » Videos

The [Videos command](#) in the [Help menu](#) displays the videos that can help you learn to use Keyboard Maestro.

Help » Open Logs Folder

The [Open Logs Folder command](#) in the [Help menu](#) displays the Keyboard Maestro Logs folder.

Help » Open Preferences Folder

The [Open Preferences Folder command](#) in the [Help menu](#) displays the Keyboard Maestro Preferences folder.

Help » Third Party Licenses

The [Third Party Licenses command](#) in the [Help menu](#) displays the licenses folder containing the third party licenses for code used in Keyboard Maestro.

Help » Online Documentation

The [Online Documentation command](#) in the [Help menu](#) takes you to the Keyboard Maestro web site and displays the documentation.

Help » Keyboard Maestro Web Site

The [Keyboard Maestro Web Site](#) command in the [Help menu](#) takes you to the Keyboard Maestro web site.

Help » Stairways Software Web Site

The [Stairways Software Web Site](#) command in the [Help menu](#) takes you to the Stairways Software web site.

Help » Service and Support

The [Service and Support](#) command in the [Help menu](#) displays the service and support details, including how to contact us and where you can get more assistance.

Help » Report Bugs or Feature Requests

The [Report Bugs or Feature Requests](#) command in the [Help menu](#) allows you to report any bugs you find or and features you would like implemented (hat tip to the wonderful image editor, [Acorn](#)).

Status Menu

The [Status Menu menu](#) appears on the right side of the menu bar showing (by default) the Keyboard Maestro icon (although you can control the icon in the [General preference pane](#)). It is generally visible any time the [Keyboard Maestro Engine](#) is running (unless you turn it off in the [General preference pane](#)). You use it to control the [Keyboard Maestro Engine](#), to execute your Status Menu triggered macros, and to switch to applications.

Status Menu » Launch Keyboard Maestro Editor

The [Launch Keyboard Maestro Editor](#) command in the [Status Menu menu](#) launches Keyboard Maestro so you can edit your macros. You can also edit your macros directly by holding down the option key while choosing them in the [Status Menu menu](#) or in a palette.

Status Menu » Cancel

The [Cancel](#) command in the [Status Menu menu](#) contains a list of all currently running macros and allows you to cancel them.

You can also cancel all currently running macros by holding all the modifiers (Command, Control, Option, Shift) down and clicking on the [Status Menu menu](#).

Status Menu » Start Debugging

The [Start Debugging](#) command in the [Status Menu menu](#) opens the [macrodebugger window](#) and starts debugging.

See also the [Macro Debugger](#) section.

Status Menu » Quit Keyboard Maestro Engine

The [Quit Keyboard Maestro Engine](#) command in the [Status Menu menu](#) quits the [Keyboard Maestro Engine](#).

Tips

- [Remembering Macro Hot Keys](#)
- [Use the Conflict Palette](#)
- [Use the Trigger Macro by Name Action](#)
- [Use Function Keys for Global Hot Keys](#)
- [Use the Number Pad](#)

Remembering Macro Hot Keys

Hot Key Macros are only useful if you can remember which key does what.

Consider using mnemonic Macros. For example, in your email client, you might define a set of Macros to Insert Text, so use Control-A for your Address, Control-S for your Signature, Control-N for your Name, and so on.

Be consistent in your choice of Hot Keys. For example, use function keys to launch applications, Control-Function Keys to open documents, Control-Letter to Insert Text, and so on.

Keyboard Maestro also interoperates with KeyCue – if you use both applications and hold the control key down KeyCue will display all your active Hot Keys.

Use the Conflict Palette

If you have two or more macros with the same Hot Key, pressing that key will display the conflict palette, listing all the conflicting macros. You can then press further keys to filter the list until one remains (which is immediately executed).

Use the Trigger Macro by Name Action

Another way to reduce having to remember Hot Keys is to use the Trigger Macro by Name action to trigger macros based on their name.

Use Function Keys for Global Hot Keys

It is quite hard to come up with global Hot Keys that will not conflict with those keys used by any application (a conflict is not really a problem, the Macro Hot Key will simply override the application, but this is not always desirable). It is best to use function keys, especially in conjunction with modifiers, as global Hot Keys since they tend not to be used by most applications.

Use the Number Pad

Remember that the number pad is available (and distinct from the numbers on the main keyboard).

Troubleshooting

- [Macros/Switchers do not work after I login, what's wrong?](#)
- [My Macros are not working, what's wrong?](#)
- [The Window Switcher shows an empty list, what's wrong?](#)
- [The Window Switcher shows only some of my windows, what's wrong?](#)
- [The Application, Window and Clipboard Switcher do not work at all, what's wrong?](#)
- [AppleScript's display dialog does not work, why?](#)
- [I have an X-Key keyboard, are there any issues?](#)
- [How do I get more help?](#)

Macros/Switchers do not work after I login, what's wrong?

The Macros and Switchers were all working fine, but then I restarted or logged out and back in, and now they are not working any more, what's wrong?

The actions are all enabled by the Keyboard Maestro Engine. You can start it by launching the Keyboard Maestro application, or have the engine start automatically by enabling the “Launch Engine at Login” preference in the [General preference pane](#).

My Macros are not working, what's wrong?

In order for macros that use actions like Select Menu Item or Manipulate Window to work you must enable

access for assistive devices in the Universal Access system preference.

Also, check that the macro group that contains the macros is enabled for the application you are testing with.

And finally, check that the Keyboard Maestro Engine is running (the Keyboard Maestro icon should be in the menu bar unless you have turned that off or the engine is not running).

The Window Switcher shows an empty list, what's wrong?

In order for the Window Switcher to work you must enable access for assistive devices in the Universal Access system preference.

The Window Switcher shows only some of my windows, what's wrong?

The Windows Switcher can only see windows in the current Space, so if you are using Spaces, you will not see windows in another Space.

The Application, Window and Clipboard Switcher do not work at all, what's wrong?

The Hot Keys for the Application, Window, and Clipboard Switcher are all Macros in the Switcher Group, so if you have disabled this group (or restricted it to certain applications), that will affect the switcher macros as well.

Also, ensure that the Switcher Group and the macros are enabled.

AppleScript's display dialog does not work, why?

AppleScripts are executed via `osascript`, so they run in the background, do not lock up the engine, and cannot crash the engine. However this means that they cannot perform user interaction. If you need user interaction (such as `display dialog`), you can work around this by asking another application (usually System Events is a good choice) to perform the request. For example:

```
tell application "System Events"
  activate
  display dialog "Hello"
end tell
```

See also the Scripting section.

I have an X-Key keyboard, are there any issues?

In general, Keyboard Maestro's Device trigger can work with all the P.I. Engineering's X-Key or other programable keyboards.

To use a key, the key must act like a button, that is it must be a single switch that is pressed on/off. You cannot use jogs, shuttles, proportional joysticks or (generally) sliders as trigger buttons.

To detect a key, Keyboard Maestro must be able to see an individual key turn on and then turn off. If you are using a double key on your device, you must remove it and configure either of the single keys and then you can replace it – otherwise it will seem like you are pressing two keys and Keyboard Maestro will not know which to use.

Currently, Keyboard Maestro cannot recognise the left most column of the MWII Jog & Shuttle, the top two rows of the LCD No Reader, and the left most column of the 128-Key.

How do I get more help?

For more information about a specific Keyboard Maestro feature consult the Keyboard Maestro Documentation, post a question to the Keyboard Maestro Forum, visit the Keyboard Maestro web site or the Keyboard Maestro Wiki or contact us.

We always respond to email, however email is no longer a guaranteed medium and spam filters can delete your message to us or our message to you. Messages sent using the feedback form will always get to us, emails sent to us will pretty much always get to us, but if you do not receive a response within one business day check your spam filters to see if they have trapped our reply. If you use the feedback form and want a reply, make sure you enter your email address!

Support

For sales enquiries, customer service, technical support, or to contact project management, email us at support@stairways.com or use our [Web Site Feedback Form](#).

The documentation is by consulting the [Keyboard Maestro Documentation](#) and for up to date information, visit the [Keyboard Maestro Wiki](#).

You can join the [Keyboard Maestro Forum](#) online community consisting of the developers and Keyboard Maestro users.

For ideas, see the [Macro Examples](#) section, or the [Macro Library](#) section of the [Keyboard Maestro Wiki](#).

You can download Keyboard Maestro from <http://download.stairways.com/>. You can download old versions of our applications from the archive site at <http://files.stairways.com/>.

You can purchase Keyboard Maestro at <http://purchase.stairways.com/>.

You can look up your current or previous license status and serial numbers, and get information about discounted upgrades from <http://enquiry.stairways.com/>.

For more information about anything to do with Keyboard Maestro visit <http://www.keyboardmaestro.com/>.

Glossary

Clipboard

The system clipboard is where you store items when you Copy and Paste. When you Copy an item, it is temporarily stored in the Clipboard and when you Paste, the item is copied from the Clipboard into your currently selection.

Clipboard History

Normally the system stores only one clipboard. Keyboard Maestro keeps a history of your system clipboard, ensuring you never lose data on the clipboard and allowing you to copy and paste multiple items,

Clipboard Switcher

is a feature of Keyboard Maestro that allows you to copy or paste to/from a set of [Named Clipboards](#).

Excluded Applications

is the set of applications that should not appear in the Application Switcher list, allowing you to hide applications you rarely want to switch to. These applications are also ignored when hiding other applications.

Global Macro Group

a predefined [Macro Group](#) that always exists and is the default location for new Macros.

Growl

a system extension that lets Mac OS X applications unobtrusively tell you when things happen ([more info](#)).

Hot Key

A keystroke that acts as a [Macro Trigger](#) to start the execution of [Macro Actions](#) in a Macro.

KeyCue

software from ergonis that displays command keys and can also display Keyboard Maestro [Hot Keys](#) ([more info](#)).

Keyboard Maestro Engine

The process that enables your [Macros](#), [Application Switcher](#), [Window Switcher](#), [Clipboard Switcher](#) and web server to work even after you quit Keyboard Maestro.

Mac OS X

Apple's operating system versions 10.0 and up.

Mac OS

Apple's operating system we've all come to know and mostly love.

Macro

a set of [Macro Triggers](#) together with a sequence of [Macro Actions](#). Any one of the triggers will begin the execution of the sequence of actions.

Macro Action

an action you wish to perform, such as opening a file, typing some text, controlling iTunes, and so on.

Macro Group

a set of Macros which can be restricted to only a defined set of applications.

Macro Palette

a floating palette containing any active Macros that have a Macro Palette trigger. The palette only appears in applications with at least once active Macro Palette triggered Macro.

Macro Trigger

an event, such as a [Hot Key](#), application launch, time of day, that starts the execution of a Macro.

Michael Kamprath

the original developer of Application Switcher and Keyboard Maestro.

Named Clipboard

Keyboard Maestro provides a set of named clipboards where you can permanently store information (text, logos, graphics, etc).

Notification Center

Mac OS X Mountain Lion introduced a system wide notification center which shows you alerts in the upper-right corner of your screen, without interrupting what you're doing.

Program Switcher

the premier application management utility for Classic Mac OS, written by [Michael Kamprath](#) it was in part the inspiration for Keyboard Maestro and forms one of the components of Keyboard Maestro.

Quick Macro

a macro recorded on the fly in another application. see the [Recording](#) section.

Record Quick Macro

the action that when triggered records a [Quick Macro](#). see the [Recording](#) section.

Regular Expression

a way of matching strings based on patterns. When Keyboard Maestro refers to “matching” it means by regular expression, in particular [ICU Regular Expressions](#).

Sandboxed

an Apple technology that limits what an application can do and how it can interface with the rest of the system and other applications. This is great for security for applications that work with just their own data, but is impossible with workflow applications like Keyboard Maestro, Automator and AppleScript Editor.

Shortcut

a Shortcut is another name for a [Macro](#) (it is also another name for an Alias but that is a different context to the normal Keyboard Maestro Macro context).

Text Tokens

text tokens allow you to insert dynamic text, such as the current date or time, into various text fields, see the [Text Tokens](#) section.

Tokenized

see [Text Tokens](#).

UID

a unique identifier that remains the same even if you rename or modify a macro or macro group.

Z-order

refers to the order of windows from frontmost to furthest back.

Administrative Details

- [Requirements](#)
- [Distribution](#)
- [History](#)
- [Credits](#)
- [Warranty](#)
- [Licenses](#)
- [Fine Print](#)

Requirements

Keyboard Maestro 6 requires [Mac OS X 10.8](#) or later.

Distribution

You may distribute this application in any way you wish as long as you only distribute the unmodified Keyboard Maestro package, as downloaded from www.stairways.com. You may not break Keyboard Maestro up into its component files and distribute parts of it separately.

History

Following on the success of Application Switcher for Classic Mac OS, [Michael Kamprath](#) wrote Keyboard Maestro for [Mac OS X](#) and released it in early 2002. Incorporating an impressively powerful hot key macro facility, as well as Application and Clipboard Switching facilities, it rapidly became an indispensable tool for many [Mac OS X](#) users, including us here at Stairways Software.

Development continued on version 1 through the end of 2002, and then work began on version 2. The first beta of 2.0 was released in early 2003 and development continued until the 2.0b6 beta released in May 2003. After that, life and work got in the way. Keyboard Maestro languished for over a year as Michael found that he did not have the time or energy to continue development.

Around May 2004, we contacted Michael as a concerned user to query the long delay in the eagerly awaited 2.0 release. When we learned that Michael was considering abandoning the application we offered to

purchase it from him to ensure that we would not lose this valuable tool, as well as to continue the fine tradition that he had started.

On June 30, 2004 the deal was struck and Stairways Software acquired all the rights to Keyboard Maestro. Our aim was to resolve the outstanding issues with Keyboard Maestro and release 2.0 as soon as possible, which we did in September 2004. Keyboard Maestro 2 introduced many new [Macro Triggers](#) (such as Application, Time of Day, and so on), [Macro Groups](#) to allow easy control over when macros are active, and many new actions.

Development of Keyboard Maestro competed for resources with development of Interarchy until the latter was sold to lead developer Matthew Drayton in early 2007. After a short break, development on Keyboard Maestro 3 started in earnest and resulted in many new features, including improved and streamlined user interface, recording, new triggers, built-in web server, new actions, and numerous minor enhancements. Keyboard Maestro 3 was released in April 2008 followed by a succession of releases over the rest of 2008.

Development of Keyboard Maestro 4 began in late 2008 and was released in late 2009. Version 4 was a complete rewrite of the user interface, bringing with it a modern look and feel reminiscent of various modern Apple applications. Further minor releases were made through 2010, followed by the initial release of Keyboard Maestro's baby brother Switcher Maestro and the Mac App Store version in January 2011.

By that point, development of Keyboard Maestro 5 was well under way and was released in July 2011. Keyboard Maestro 5 built on the solid user interface of version 4, but added depth and breadth of power with almost no addition of complexity. Keyboard Maestro added such powerful features as control flow, conditions, variables, and calculations as well as many new actions, and enhancements to the application and clipboard history switchers. Further minor releases were made over the next year, adding things like a For Each action, File actions and Image actions.

Development of Keyboard Maestro 6 began in mid 2012 and was released in May 2013. Version 6 kept the user interface largely unchanged, while adding significant new features, including macro syncing, macro debugging, plug in actions, full support for styled text, support for controlling Safari and Google Chrome web browsers, trigger by name facility. Keyboard Maestro 6 also sported a stylish new icon from [Iconaholic](#) as well as customizable status menu icons and full Retina graphic support.

Going forward, we plan to develop Keyboard Maestro aggressively, bringing it to new levels of both power and ease of use in the long tradition of both [Mac OS](#) and Stairways Software.

Credits

Thanks to [Michael Kamprath](#) for all his work producing Keyboard Maestro.

Thanks to Alan Gentle for many example Macro ideas.

Thanks to Philippe Martin for some great beta testing.

Thanks to [Dan Benjamin](#) for doing the voice overs on the tutorial videos.

Thanks to [Noah Kadner](#) the voice overs on the intro video.

Thanks to Rakesh Kumar for the set of Switcher Macros.

Thanks to Sam Stephenson and the Prototype Core Team for the Prototype JavaScript Framework.

Thanks to Jono Hunt for the brilliant [Iconaholic](#) icon and other help.

Thanks also to:

- [Jerry Krinock](#) for NS(Attributed)String+Geometrics.
- [John Gruber](#), [Aristotle Pagaltzis](#) for Title Case.
- [Kelan Champagne](#) for the [YRKSpinningProgressIndicator](#).
- [Matt Gemmell](#) – Magic Aubergine for MGAnimatedView and MGTemplateEngine.
- [Michael Ash](#) for MAKVONotificationCenter.
- Matthew Ball for [MBCoverFlow](#).
- [Rainer Bockerhoff](#) for [RBSplitView](#).
- [Keith Blount](#) for KBWebArchiver.
- Random Ideas for NSTimer+Blocks.
- [Willow Garage](#) for [OpenCV](#).

Thanks to Ken, Corentin, Stephen, Brad and others for their great assistance with beta testing.

Thanks to Andy for great help editing this documentation.

Thanks also to the many others who have provided input and support over the past decade.

Warranty

This application should do what we have described in this document. If it does not, you can simply stop using it. If you purchase it, and within 30 days find that it does not do what we have described here, then you can request a refund and your money will be refunded and we will cancel your license.

Licenses

Keyboard Maestro is copyright 2015 Stairways Software Pty Ltd. All Rights Reserved. You may use this application for a short trial period and then you must purchase the application or stop using it.

Keyboard Maestro is licensed on a per user basis and individual users may use it on up to five Macs. You must purchase a license for each user using Keyboard Maestro.

Trademarks owned by Third Parties such as Mac, Mac OS X, and BBEdit, are owned by their respective owners and no license is granted for their use.

Fine Print

Keyboard Maestro, Switcher Maestro, keyboardmaestro.com, switchermaestro.com and stairways.com are the property of Stairways Software Pty Ltd. Stairways Software Pty Ltd hereby disclaims all warranties relating to this software, whether express or implied, including without limitation any implied warranties of merchantability or fitness for a particular purpose. Stairways Software Pty Ltd will not be liable for any special, incidental, consequential, indirect or similar damages due to loss of data or any other reason, even if Stairways Software Pty Ltd or an agent of theirs has been advised of the possibility of such damages. In no event shall Stairways Software Pty Ltd be liable for any damages, regardless of the form of the claim. The person using the software bears all risk as to the quality and performance of the software.