Ukelele 3.0

Unicode Keyboard Layout Editor

User’s Manual

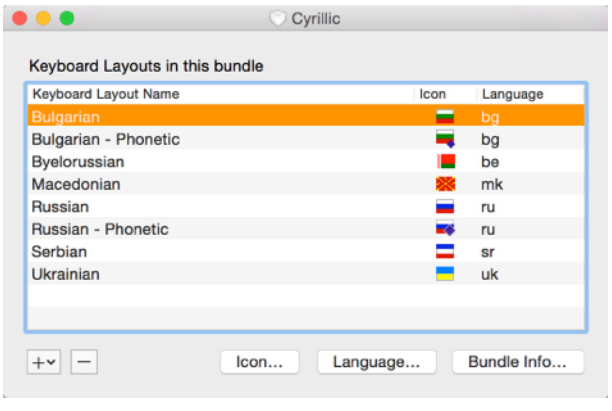


Table of Contents

1. Introduction

1.1. What is Ukelele?

Since at least Mac OS 8.5, applications have had access to Unicode, the international standard encoding for characters in most languages of the world. The main difficulty in using Unicode has been how to enter the characters you want. In Mac OS 8.5, Apple introduced an alternative to the earlier type of keyboard layout, which could only handle the various scripts defined by Apple (MacRoman, Cyrillic, and so on). These keyboard layouts were all but impossible for users to change.

With Mac OS X 10.2, Apple introduced a new type of keyboard layout, an XML file. This allows users to make whatever changes they want, so that they get exactly the keyboard layout they need. However, editing XML is tedious and error-prone, and it is often very difficult to work out what has gone wrong when the keyboard layout doesn't work.

Ukelele is designed to create and edit these XML keyboard layout files, using a simpler, graphical interface, so that creating custom keyboard layouts is possible for ordinary users.

 Its web site is http://scripts.sil.org/ukelele. Ukelele 3.0 is compatible with Mac OS X 10.8 (Mountain Lion) and later. Older versions are compatible with Mac OS X 10.4 (Tiger) and later (version 2.x) Mac OS X 10.2 (Jaguar) and later (version 1.x).

1.1.1. What Ukelele can do

Ukelele can create and edit almost any kind of keyboard layout that is in the appropriate XML format. You can modify the keyboard layout to produce any Unicode character, or even short strings of Unicode characters. Do you need a keyboard layout for Ethiopic, or Devanagari, or for producing mathematical or phonetic symbols? You can create one with Ukelele.

Almost all the keyboard layouts provided by Apple are available as XML versions, or can be converted to XML, so that you can create a new keyboard layout based on one of them. Do you wish that two keys on a particular keyboard layout were swapped? You can do that with Ukelele.

You can create complex dead keys that turn a series of key strokes into the desired output, as ⌥u followed by a vowel produces the vowel with a diaeresis in the US keyboard layout. These are often used with languages that have multiple diacritics, such as breathing marks and accents in Greek, or tone marks in many languages.

1.1.2. What Ukelele can't do

Ukelele is not intended for remapping modifier keys. For example, it is not intended for swapping the command and option keys. Mac OS X 10.4 introduced some flexibility in rearranging the modifier keys. Otherwise, other software, such as Karabiner (formerly KeyRemap4MacBook)

What is Unicode?

Unicode is an evolving standard for specifying computer codes for virtually every character you might need. A character can be a letter such as a Latin (or Roman) h, a Cyrillic Ⱨ, a Greek π, an Arabic ص, a Hebrew ע, a Gurumkhi ਣ, a Japanese ク, a Korean 뻥, a Chinese 哳, or even a letter from an archaic script like Ugaritic. It can also be a symbol, such as punctuation marks (-,.—&;), phonetic symbols like ɖ, mathematical symbols like ∃, musical symbols like ♭, arrows (➔➴↴⇦), currency symbols like ¥, chess piece symbols like ♗, astrological symbols like ♆, and many others.

Up to the 1990s, there were various standards around. The long-time standard of ASCII only defined a simple collection of Latin letters, numbers and punctuation. Beyond that, there were different ways to extend it. The Mac had an approach where the font would define the characters you wanted, and you had to type sometimes arbitrary characters to get the symbols you wanted, particularly with dingbat fonts. There was a standard for putting Latin letters which weren't defined in ASCII, which was called MacRoman. The Windows world had several such standards, ANSI, Windows-1252, ISO-8859-1, and so on.

This led to conversion problems that you have most likely seen, often in email, where curly quotation marks will appear as something else, such as í. These problems were painful to deal with, and approaches like the Text Encoding Converter were created to help with them.

Unicode gets around the problem by defining a single code for each character. In the current standard (8.0), there are over two million possible codes, though not all codes get a character, as there are some gaps to make new blocks of characters start at multiples of 16. Characters are guaranteed to keep their codes over the life of the Unicode standard, with new characters being added rather than moved around or deleted. So, when you have your text in Unicode, it should look the same no matter what computer reads it — Mac, Windows, Linux, or anything else, as long as it understands Unicode, which most modern applications do, and the fonts include your characters.

(http://www.macupdate.com/app/mac/25141/karabiner) or DoubleCommand (http://doublecommand.sourceforge.net/), can do this and more.

Ukelele cannot turn a modifier key into a non-modifier key or vice versa. A modifier key is one of shift, option, control, command, caps lock and fn. Again, DoubleCommand can do some of this, such as turning the Enter key into a command key.

The fn key is not a modifier in the same sense that the shift key is. The fn key really acts as a way for a notebook keyboard (as in PowerBook, iBook, MacBook or MacBook Pro) to simulate a full-sized keyboard with a separate row of function keys (F-keys, F1–F15 or F16, or even F19) and a numeric keypad. This effectively limits the fn key to working with only a subset of the keys on the keyboard. It gets even worse with more recent models of MacBook and MacBook Pro, which do not have a full embedded keypad, so that the fn key only works for turning return into enter and allowing access to the F-keys.

There are some keys which perform hardware functions such as controlling sound volume or display brightness. Ukelele cannot change these functions.

There are some limitations imposed by Apple:

- Mapping the arrow keys to something else, or other keys to arrow keys, is problematic. It will work in some applications, but not others.

- Control key combinations often do not work as desired. Often, control key combinations are simply ignored, and sometimes they do something unexpected. There are some ways around some, if not all, of the limitations, but these are not for the faint of heart!
- Input methods, such as Chinese, Japanese or Korean, will not work with modified keyboard layouts.
- A single key cannot produce more than 20 Unicode characters as output.

1.2. The name Ukelele

Yes, we know that the more "correct" name of the musical instrument is "ukulele". However, "ukelele" is a well-accepted spelling in at least Australia and the UK. The name came from Unicode Keyboard Layout Editor, which gave UKLE, and the rest is the responsibility of the author.

1.3. Help

There are several sources of help. One is this manual. Another is a help book accessible from within Ukelele, via the Help menu. There is a (defunct) forum for comments on the Ukelele home page, but it is better to use the Ukelele Users group on Google Groups, which allows uploading of files, a mailing list and other information.

2. Installation

Ukelele is self-contained, and you only need to drag and drop the Ukelele application from the disk image into the Applications folder on your computer, or wherever you want it. To uninstall Ukelele, remove the application from your computer. Only two other files are created, in the preferences folder of each user. One is a file called org.sil.ukelele.plist, which contains Ukelele's preferences. The other is a file called org.sil.Ukelele.LSSharedFileList.plist, which contains things like recent documents opened by Ukelele.

One exception is that using Ukelele to install a keyboard layout creates a helper tool which runs in the background. If you want to remove this, there is a tool on the disk image called Uninstall Helper Tool.sh. You need to run this within a terminal window, and will have to authenticate as an administrator for your computer.

Also on the disk image are the Read Me file (which gives the update history), the manual (in PDF format), and three folders, one called System Keyboards, one called Logitech Keyboard Layouts, and the other called Tutorial. The System Keyboards folder contains a collection of keyboard collections, each containing a set of keyboard layouts and their associated flag icons. These are XML versions of the keyboard layouts that were available on Mac OS X 10.4, and are intended as starting points for creating new keyboard layouts. The Logitech Keyboard Layouts bundle contains XML versions of keyboard layouts supplied with the Logitech Control Centre, obviously intended for Logitech keyboards, but useful for others, as well. The Tutorial folder contains a tutorial introduction to Ukelele. This tutorial is dated, meant for Ukelele 2.x, but many things will be similar. Open the index.html file in a web browser to run the tutorial.

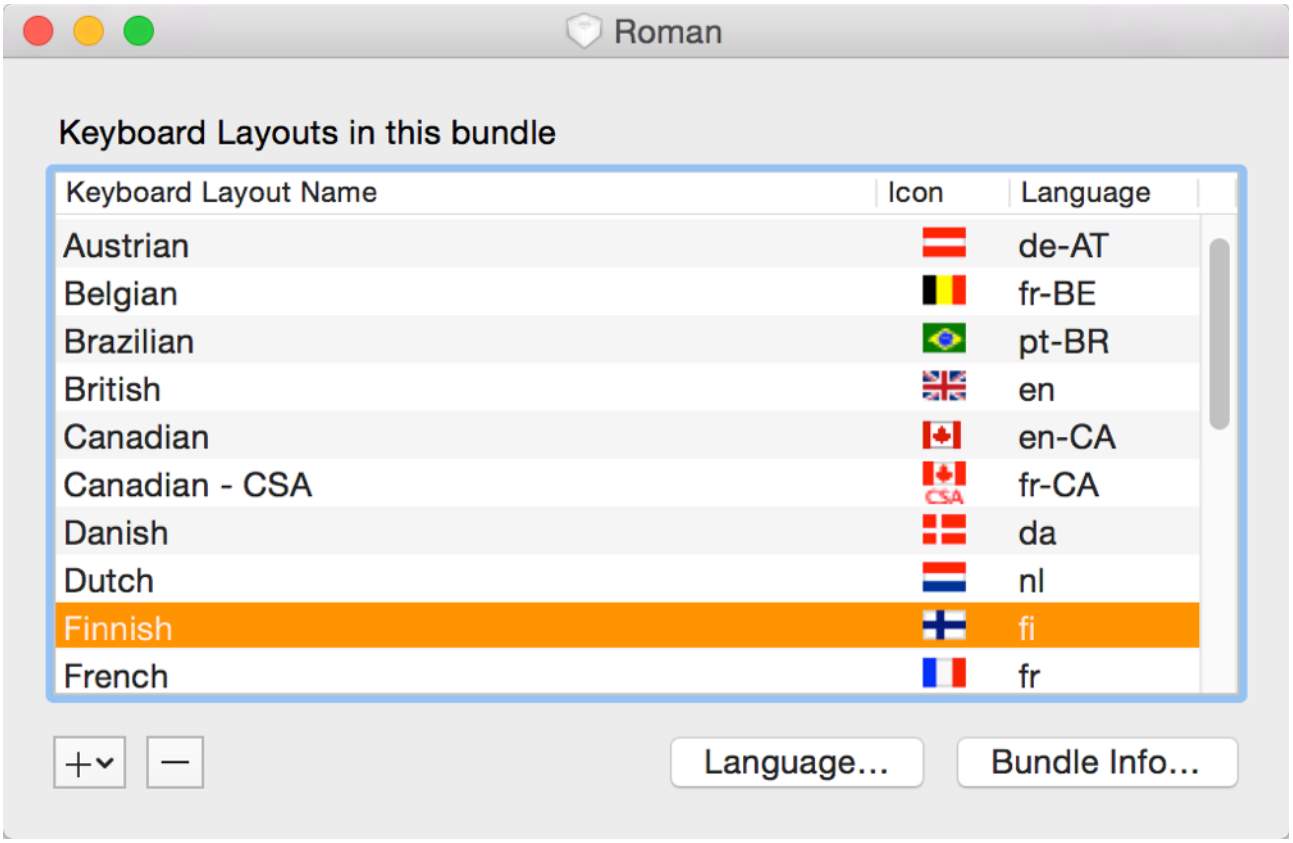
3. Quick Start

This chapter gives a brief overview of using Ukelele. Fuller discussion is in chapters 5, 6 and 7.

3.1. Making changes to a keyboard layout

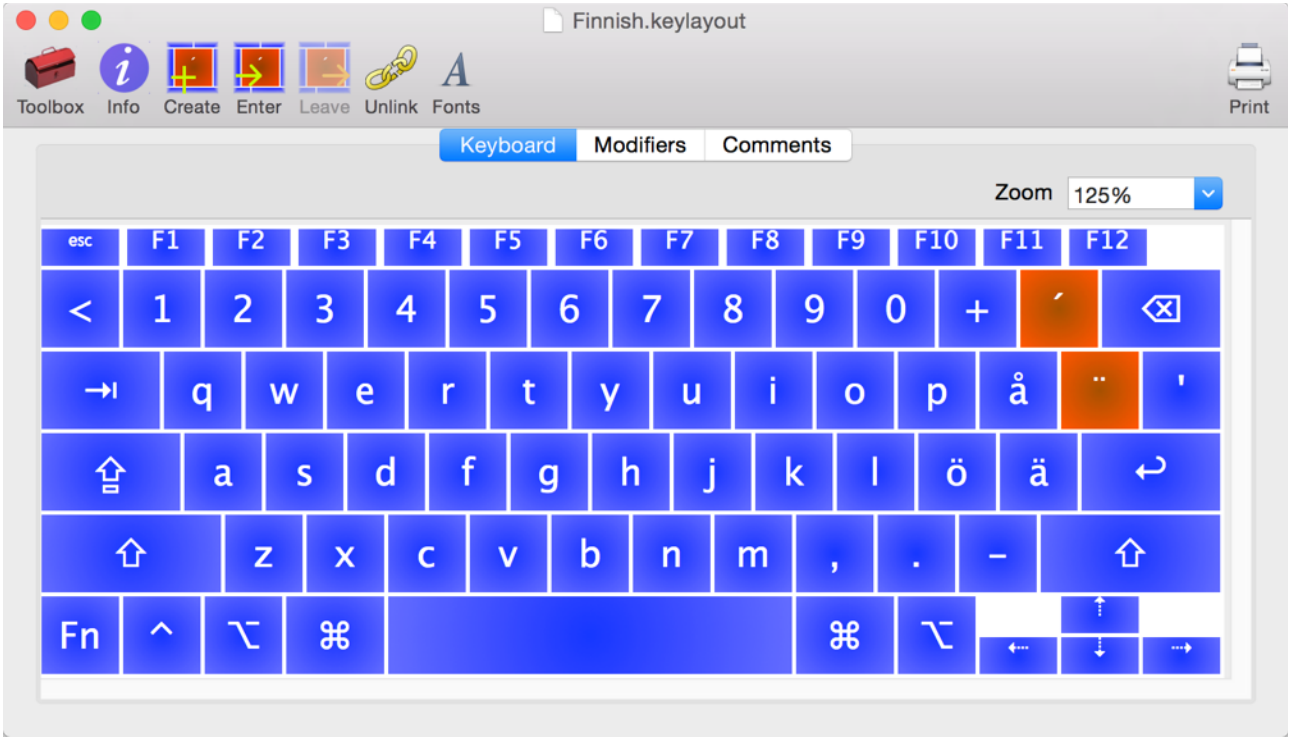
Making changes to a keyboard layout can be very simple, but there are more complex things such as dead keys to consider. To begin with, we look at how to change the output of a key.

Open Ukelele, by double-clicking the icon or any other standard method. Now, choose Open... from the File menu, and choose an existing keyboard layout. For the purposes of this section, we will choose the Roman document in the System Keyboards folder provided with Ukelele. You will see a window like this:



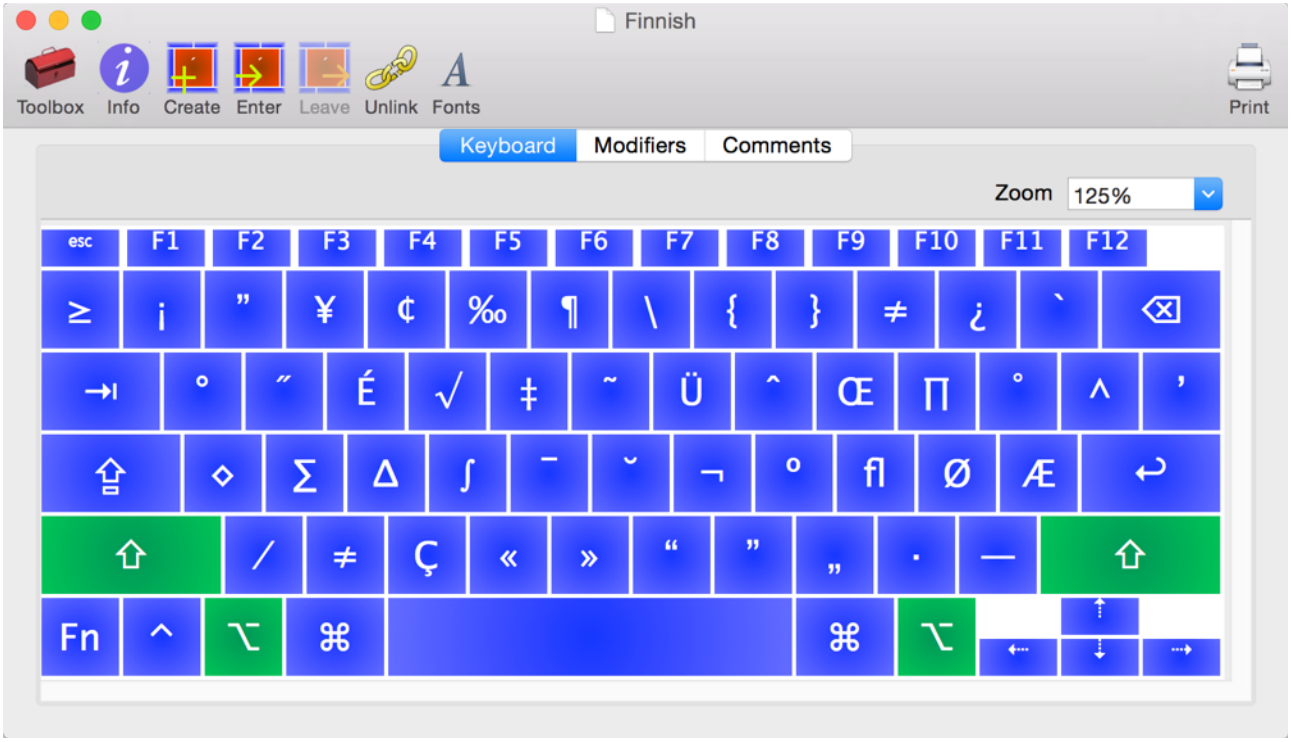
Select the Finnish keyboard layout from the list, and either double-click it or right-click on it and select Open Keyboard Layout.

You will see a window like that shown in the figure. It may look somewhat different, depending on what hardware keyboard is attached to your computer. Try pressing various keys on the keyboard, and observe that the corresponding key in the window changes to indicate that it has been pressed. Pressing on one of the modifier keys (shift, caps lock, option, command, or



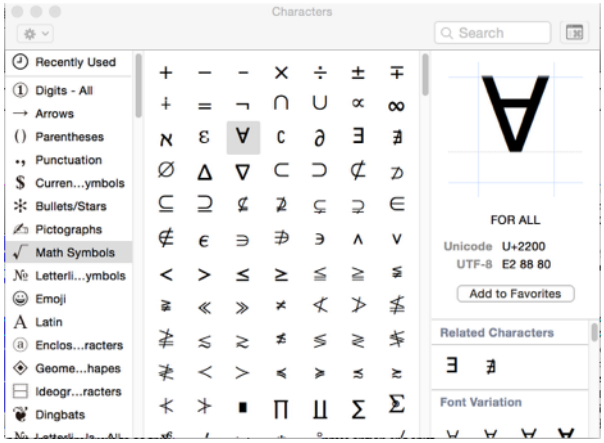
control) changes the keys to show what output you would get from the keyboard layout with that modifier down.

If you hold down shift and option, you will see that shift-option-2 and shift-option-m both produce the same character, a double right quotation mark, ”. What we'll do now is to change shift-option-2 so that it produces another character that currently isn't on the Finnish keyboard layout, the mathematical symbol “for all”, ∀.



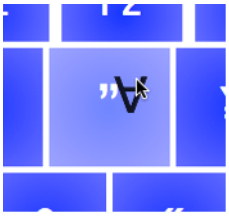
There are several ways to do this:

3.1.1. Drag and drop



If the Character Viewer is not open, open it from the input menu, which is a flag towards the right hand side of the menu bar. If you don't have an input menu, or the character viewer is not listed there, open the Keyboard pane of System Preferences and turn it on. This is different in different versions of Mac OS X. If you're stuck, see the details in chapter 5.

In the character viewer, find the Mathematical Symbols section, and find the ∀ character. Now, hold down shift and option, so that you can see the ” on the 2 key. Select the ∀ character in the character palette, and drag it onto the 2 key. (You may have to click on the title bar of the Character Viewer to enable dragging.) Note that the key is highlighted with a lighter colour to indicate that you can drop the character onto the key. You should now see that the 2 key shows the for all symbol when shift and option are held down.



If you don't want to use the character viewer, you can use this manual. Just select the symbol, and drag it into the Ukelele window. It works with any application that understands Unicode and supports drag and drop. The character palette is simply an easy way to get at the full range of Unicode characters.

3.1.2. Double-click

To use the second method, you hold down option and shift, and double-click the key in the Ukelele window. This brings up a popover that shows you the output that is currently associated with the key, and allows you to change it. Note that the old output (possibly represented as an XML entity) is already selected, so inserting the new output will erase the old.

Again, we will use the character viewer. If it is not showing, open it from the input menu. Locate the for all symbol and double click on it (or click the Insert button if there is one). You should see that the old output is replaced by the for all symbol in the dialog. Click Done, and that's it.

If you don't use the character viewer, you can copy the symbol from the manual and paste it into the dialog and click Done.

3.1.3. Entering code points

The third method is a variation of the second. Hold down the option and shift keys, and double-click the key in the Ukelele window. Now, we will need to know the Unicode code point for the for all symbol. It is hexadecimal 2200, or decimal 8704. We have to enter the XML code for a character specified by code point. This is either ∀ or ∀. Note the ampersand followed by a hash mark at the beginning, and the semicolon at the end. If there is an x after the hash mark, the rest is interpreted as hexadecimal, otherwise it is interpreted as decimal. Enter either of those codes into the dialog and click OK. You should now see the for all symbol when you press option and shift.

3.2. Creating a new keyboard layout

To create a new keyboard layout, you have three basic choices. You can create a standard keyboard layout, create a keyboard layout based on an existing keyboard layout file, or you can capture the current keyboard input source as a keyboard layout.

3.2.1. Standard keyboard layout

There are various standard keyboard layouts available. These include an empty keyboard layout (no output defined for any key) and various standards for different languages: QWERTY, Dvorak and Colemak for English, QWERTZ for German, and AZERTY for French. To create an empty keyboard layout, choose New from the File menu, or press ⇧⌘N.

An empty keyboard layout is not truly empty, but contains output for a few basic keys, plus a simple set of modifier combinations (none, shift, option, caps lock and shift-option). However, most of the keys do not have any output attached to them, so that most keys look blank. From there, you can create whatever you want the keyboard layout to be. Be aware that it will be a long and somewhat tedious task, so that you normally only use this option if there is no similar keyboard layout to base yours on.

3.2.2. Based on an existing layout

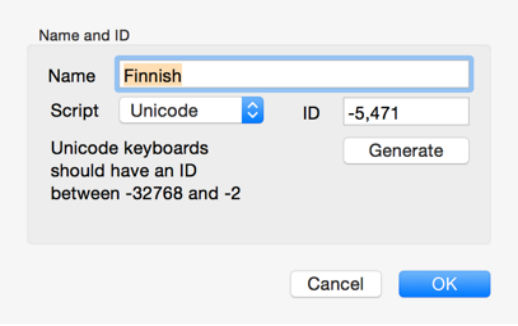
The most common option is to create a keyboard layout which is like another keyboard layout. Choose Open... from the File menu (or press ⌘O), and Ukelele will bring up a standard open file dialog. From this, you can choose the keyboard layout on which you wish to base your keyboard layout. You should immediately choose Duplicate from the File menu (or press ⇧⌘S), which will create a copy of the keyboard layout which you can save as your new keyboard layout.

3.2.3. Capture the current keyboard input source

If you have a keyboard layout for which you do not have a keyboard layout file available, but do have some other way of using it, such as an old resource-based keyboard layout or a system keyboard layout, you can convert this to a keyboard layout that is editable within Ukelele. Choose New From Current Input Source from the File menu, and Ukelele will do the conversion for you and create a keyboard layout that you can edit and save.

3.3. Installing and using a keyboard layout

Before you are done with a new keyboard layout, it is important to give it a name that you will recognise. This is not the same as the file name where it is saved, but a property of the keyboard layout itself. Select Set Keyboard Name and ID... from the Keyboard menu, and you will see a dialog like this:



Set the name you want, and ensure that the right script (almost always Unicode) is set, and that the ID is in the correct range.

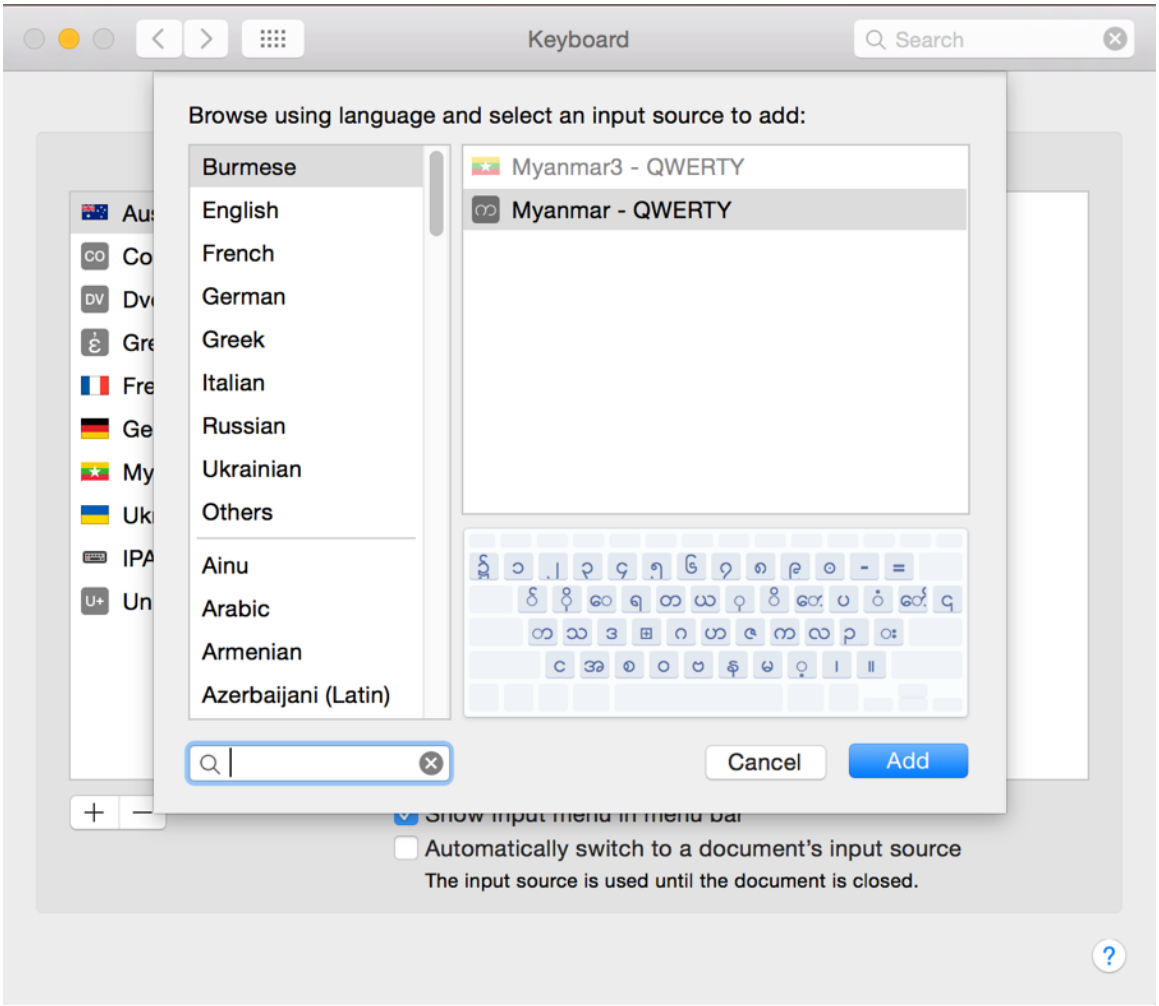
Once you have a complete keyboard layout, you need to install it. The simplest way to do this is with Ukelele itself. You can choose to install for the current user, or for all users on your computer (which requires an administrator password). These commands are found on the File menu, Install submenu.

Once the keyboard layout is installed, it has to be activated. This is done in the System Preferences application. You select the Keyboard pane, or Language & Text on OS X 10.8 (Mountain Lion), and select the Input Sources tab. What happens next depends on your operating system.

On OS X 10.8 (Mountain Lion), you need to check the box to the left of the keyboard name.

On OS X 10.9 (Mavericks) and later, click the + button below the list of active keyboard layouts. This brings up a list on the left of languages, including "Others", which allows for languages not listed by the system. Choosing a language gives you a choice of keyboard layouts appropriate for that language. Clicking the Add button activates the keyboard layout.

In any system, the keyboard layout name is that set in the dialog earlier, *not* the file name.



After activating the keyboard layout, you should be able to choose it from the input menu, which normally has a flag icon, towards the right of the menu bar. You can now type with your keyboard layout.

3.4. Tutorial

There is a tutorial in the Tutorial folder of the disk image that Ukelele came on. Open this in a web browser, and it will lead you through the steps to create a functioning keyboard layout. It demonstrates several of the techniques mentioned in this chapter, and also introduces you to dead keys.

Do note that it was made with earlier versions of Ukelele, so not everything will work as depicted in the tutorial.

4. What's New in Ukelele 3

Ukelele 3 is a major upgrade from version 2.x. The most significant change is that the entire user interface has been rewritten. (For technically-minded readers, it's now a Cocoa application rather than Carbon, as earlier versions were.) This has involved a whole lot of new approaches to how various tasks are achieved, so many things are done differently.

The following list explains the main changes that affect the user:

- The new document model introduced by Apple has been adopted. This means that all changes are automatically saved rather than requiring the user to explicitly save periodically. To make a new copy of a keyboard layout, you use the new Duplicate... item in the File menu, and save the duplicate with the new name.
- Ukelele now handles keyboard layout collections (bundled keyboard layouts) much more easily, and these are the usual way that keyboard layouts are saved. Collections can have more than one keyboard layout within them, and you can easily associate icons and intended languages with keyboard layouts within collections.
- The main keyboard layout window now uses a tabbed interface to display the keyboard, modifiers and comments, rather than using a drawer for modifiers and the somewhat confusing toolbar button to switch between keyboard and comment views.
- Ukelele now has the capability to install keyboard layouts itself. You can install for yourself, or for all users on your computer, without having to save the keyboard layout and drag it into the correct folder in the Finder.
- The information inspector has been redesigned. Among other improvements, there is now a pair of fields which show which modifiers are currently pressed and which modifier set matches that modifier combination.
- A concept of "simplified modifiers" has been introduced. Since no current Apple keyboard, or any Cocoa-based application, distinguishes between left and right shift, option or control keys, treating them all as the left modifier key, it is only worthwhile using left keys. Ukelele can simplify the modifier combinations defined by a keyboard layout to eliminate use of separate left and right modifier keys. This makes the modifier combinations easier to understand, without losing anything in current Apple systems.
- Creating a new keyboard layout now uses a dialog that will allow you to create a basic layout of various sorts (QWERTY, QWERTZ, AZERTY, Dvorak or Colemak), and to have a different layout with either the caps lock key or the command key, or even a different one with each.
- Some interactions have been simplified by introducing the idea of selecting a key and then operating on it. So, for example, you could select a key and then turn it into a dead key with a single click and a single dialog.
- A QuickLook generator has been added, which means that you can see a keyboard layout from Finder by QuickLook (pressing space, choosing the contextual menu item, etc). Sadly, this doesn't work for collections, as they get the generic bundle preview.
- Editing key output is now (optionally) done with a popover, which is hopefully easier to use.

- Many dialogs have been redesigned, and some have been combined, so that some tasks are now done with a single dialog rather than a series of dialogs.
- It's now possible to use drag and drop within or between keyboard layouts within the Ukelele keyboard layout window.

5. Keyboard Layouts

This chapter deals with some of the technical issues surrounding keyboard layouts. Although this is not about how to use Ukelele, understanding this background should make working with Ukelele and keyboard layouts make more sense.

5.1. How a keyboard layout works

When you type a key on your hardware keyboard, what happens? As far as the user is usually concerned, the important thing is that the computer realises that a particular key has been typed, and it puts the appropriate character in the current input field (a document, Spotlight search field, text box in a dialog, etc). However, to do that requires a fair bit of work "under the hood".

The first thing that happens is that the keyboard sends a message to the computer that a particular key has been pressed. It tells the computer the hardware key code of the key, and what modifier keys were down when the key was pressed. Down in the central part of the operating system, the kernel, this gets transformed to an event which gets sent to the current application. Actually, there are two different kinds of events here, the "raw" key event and the text input event.

If you are playing a game or something like that, you would usually be using the raw key events. When these are in use, it is the hardware key that the user has pressed that is the important thing. What characters that key produces are very much a secondary thing here. In many cases, the current keyboard layout is not consulted at all. It's more along the lines of, "The user pressed the key with key code 36, which means fire the cannon."

The more common case is when the user wants to enter some text somewhere, and here is where the current keyboard layout comes into play. The operating system has been observing what keys have been pressed, and what the state of the modifiers is (including things like caps lock and num lock, which don't have to be pressed at the same time as the other key), and so is able to handle sequences of keys. This is what enables dead keys, of which more in a later section.

Anyway, what happens is that the operating system looks at the current keyboard layout, and decides what character or characters should be produced by the key that the user pressed. So, for example, it might recognise that the key with virtual key code 21 was pressed with the shift key held down, with the current keyboard layout being U.S. Extended, and produce the character "$".

5.2. Unicode and other scripts

Before Unicode was created as a standard, there was a fairly large number of different ways to encode different scripts. At the base was ASCII, which defines codes for 95 characters plus 33 "control codes" such as tab, delete, escape, carriage return, end of transmission, and other non-printing characters. ASCII was a 7-bit code, and left 128 possible codes for other characters in an

8-bit code. Thus were born many different codes, such as MacRoman, ANSI, Windows-1252, ISO-8859-1, GB2312, and many more.

On the Mac, the Script Manager provided a smaller number of these encodings, giving Roman, Central European, Cyrillic, Japanese, Korean, Traditional Chinese and Simplified Chinese. The reason that these are still important is that keyboard layouts can generate Unicode or one of these scripts.

There are still applications (though the number is small and decreasing) that do not understand Unicode, and so depend on the script that the keyboard layout produces. If a keyboard layout is set to produce Japanese, for example, then it should not produce non-Japanese characters, such as Cyrillic. Non-Unicode applications will not know what to do with these characters, and will likely produce question marks or boxes when characters outside the script are produced.

Ukelele is able to create and edit keyboard layouts which produce Unicode or any of the scripts supported by Apple. In most cases, you would be setting them as Unicode, but there are some uses for the scripts, since such keyboards will work with non-Unicode applications.

5.3. Input methods

Apart from keyboard layouts, you will discover that there are several "input methods" in the input menu, or the Input Sources (Input Menu on Leopard and earlier) tab of the Language & Text (Keyboard on Mavericks, International on Leopard or earlier) pane of System Preferences. These include ways to produce Chinese, Japanese and Korean (often collectively known as CJK), but also Murasu Anjal Tamil and possibly others. These differ from keyboard layouts in that they usually use some sort of secondary window for assembling a character from a series of key strokes.

The reason that input methods are worth mentioning is that there is apparently no way to customise them. Editing a keyboard layout that is used in conjunction with one of the input methods and installing it does not seem to work. As far as I know, this is due to the way that the input methods work, and there is little that can be done about it.

5.4. Dead keys

Dead keys are a powerful concept. The essential thing is that pressing a dead key produces no output by itself, but tells the operating system to treat the next key differently, and to produce something different from what it would otherwise produce. Note that they function in a way that is opposite to the analogous feature in Windows: On the Mac you type the dead key and then another, whereas on Windows you type a normal key and then a dead key.

The most common use, in Roman scripts at least, is to use dead keys to produce accented letters. If you had to provide keys for á, à, ä, and â, as well as for all the other vowels with these accents, you would quickly run out of keys. Dead keys provide a mechanism to reduce the need for all those keys. So, for example, the standard US keyboards use ⌥e followed by a vowel to produce the vowel with an acute accent, while ⌥` produces the grave accent, ⌥u the diaeresis, and ⌥i the circumflex.

In computer science terms, dead keys operate as a finite state machine (or finite state automaton). Key strokes take you from state to state, with the last producing output and returning you to the original state. Hence the term "dead key state" has been used for the system.

Apple's convention is that the starting state is called "none". When you type a dead key, you go to another state. State names are arbitrary strings, but it is often helpful to give them names that are helpful for remembering what they do. For example, ⌥e might initiate state "acute".

The most common types of dead keys are fairly simple, just modifying the next key stroke, producing output and returning to state "none". However, it is possible to have a second dead key triggered after the first, so that you go to a second dead key state. There is no limit to the length of such a chain of dead keys, but it is rarely practical to go beyond two or three.

One example where you might have chains of dead keys is where you need to specify two features which occur independently, such as length and tone. You could then have one dead key that puts a macron over the following vowel to show that it is long, and another dead key that adds a tone mark to the following vowel. You could then combine these to produce both a macron and a tone mark, which would mean two dead keys in a row.

Another example would be in typing polytonic Greek, where you would need to have both breathing marks over initial vowels, and accents as well, so that you would have one dead key indicating a smooth breathing, another indicating a rough breathing, and a set of dead keys for the different accents. Typing a word with both a breathing and an accent on the initial syllable would require typing two dead keys in sequence.

Another important concept with dead keys is the "terminator". Consider the case of ⌥e producing an acute accent. What happens if the user types ⌥e followed by a digit? We don't expect to get an accented digit, so the system does something different. Each dead key has a special string called the terminator, and it will produce the terminator when nothing is defined for that combination. So, if the terminator for the acute state is a free-standing acute accent, ´, typing ⌥e and then 5 will produce ´5. In other words, if we don't have any sensible character to produce, we put out the terminator followed by the character associated with the key that the user typed.

Note that a terminator can be the empty (or null) string. So, if you have a dead key state with the null string for the terminator, then, when the user types a combination of the dead key plus a key that doesn't produce any output in the dead key state, it is as though the dead key was not typed at all.

5.5. Using a keyboard layout

Four things have to happen before you can use a keyboard layout. First, you have to put the keyboard layout file in Keyboard Layouts sub-folder of one of the Library folders, either within your home directory, in the top level directory, or in the Network directory. The difference is that the first enables it for the specific user, the second for all users on that computer, and the third for all users on the local network.

Secondly, you need to log out and log in again. In recent versions of Mac OS X, this step is no longer necessary, but can sometimes be useful to make things work correctly. There is an important point to note here. When you log in again, the operating system looks at the modification date of the Keyboard Layouts folder, and only re-reads the keyboard layouts if the folder has been modified since the last time you logged in. That means that if you make a change to a keyboard layout that is already installed, logging out and logging back in will not work. You will need to move the keyboard layout file out of the folder and then put it back in before logging out. Logging in will then make the operating system read the keyboard layouts from that folder. Alternately, restarting the computer will always force the system to read all the keyboard layouts in the appropriate folders.

Thirdly, the keyboard layout needs to be activated. In the Keyboard (or Language & Text on 10.6 through 10.9, or International on 10.5 or earlier) pane of System Preferences, on the Input Sources (or Input Menu on 10.5 or earlier) tab, you need to activate the keyboard layout. This is done differently in 10.10, where you need to add it via the + button. On earlier systems, you check the check box next to the keyboard layout's name. Once this is done, the keyboard layout will appear in the input menu, and so it can be selected.

Finally, you have to select the keyboard layout from the input menu to make it active. Note that the selected keyboard layout is liable to change when you change applications. Partly, this appears to be a bug in Mac OS X from at least 10.2 to 10.4. However, switching to an application that does not handle Unicode will force the system to switch to a non-Unicode keyboard layout. If the application handles Unicode, but still changes to a different keyboard layout when you switch to the application, you should always be able to change it back with the Input menu.

The system will require you to have at least one of the system's installed layouts active, so that you cannot just have user-defined keyboard layouts active. The reason for this seems to be related to the idea of having a well-defined fallback for various situations. For some reason, the system will occasionally switch to a system keyboard layout, which can be confusing, so the input menu is useful in alerting you to this.

5.6. Support for "press and hold"

OS X 10.7 (Lion) introduced a new feature called "press and hold". This allows the user to press a key and hold it down for a short period, then a pop-up window appears, offering variations on the character. For example, pressing and holding the "e" key might produce a pop-up offering various accented lower-case e characters.

This feature is tied to language, with different languages offering different sets of variations. So, for this to be supported, the keyboard layout has to have an associated language. Apple's mechanism for this is to have the language specified in the bundle containing the keyboard layout. It is not currently possible with keyboard layouts stored only as .keylayout files.

Ukelele has support for language marking in keyboard layout collections. There is a new menu item, "Set Keyboard Language..." which allows you to choose the language associated with the keyboard layout. It is also available by the Language... button or the contextual menu. If you don't choose a language, the current default language of your operating system will be set. Although any language and its variations can be chosen, there is little benefit in choosing a language beyond those supported by OS X, and only some regions are supported. See section 6.4.1 for more details.

6. Creating and Editing a Keyboard Layout

In this chapter, we look at how you create and edit a keyboard layout. This is a task-oriented approach to describing how to use Ukelele. Chapter 7 provides a reference, focusing on each of the windows and menus in Ukelele.

6.1. Designing a keyboard layout

The first part of creating a keyboard layout happens before you start to use Ukelele, with designing the layout. The following questions will help you think through some of the issues that need to be addressed in keyboard layout design.

6.1.1. Who will be the users of the keyboard layout?

Is this keyboard layout one that only you will use? Do you expect other people you know to use it? Will it be made available to the wider internet community?

Answers to these will guide some choices. If this is only ever going to be your own keyboard layout, you have more freedom to do things in whatever way you find convenient. The more people that will be using it, though, the more you have to consider standards that already exist. If people from different countries are going to use it, you may have to think about how people who use a different national standard keyboard layout might react to your keyboard layout.

6.1.2. What is distinctive about the keyboard layout?

What distinguishes your keyboard layout from the standard keyboard layouts? Is it simply rearranging the keys of an existing keyboard layout, is it adding new characters to an existing layout, or is it something completely new?

In many cases, the closer it is to a standard keyboard layout, the easier it is to learn how to use it. If you change a key's output, think carefully about why you are doing it. Could there be another way to do it that might be more natural?

6.1.3. What Unicode characters will the keyboard layout produce?

What characters do you want to produce? Have you got all the ones that you or other users might need? Have you considered some of the alternate forms, such as word-final forms, precomposed characters, or combining diacritics?

If you are adding characters for a particular script, you should probably consider whether you need to add all the characters in that script. If you are only using this keyboard layout yourself, then you may not need to. If it is for wider use, you may want to add more of the characters, if not all of them.

6.1.4. Are dead keys a useful part of the keyboard layout?

Is there some sort of modifier that you would add to multiple characters, such as a diacritic? Would it make sense to change modes, say from entering Roman script to entering a Cyrillic equivalent?

6.2. Choosing a starting point

One important factor in creating a keyboard layout is your starting point. Your answers to the questions in the previous section will help here.

Your options are quite broad at this point. However, the simplest way is often to start from an existing keyboard layout which is almost what you want. If your aim is a keyboard layout which is the same as some standard, but with some changes or additions to suit your needs, then you can often find a standard keyboard layout that you can use as a base. So, for example, if you want to create a keyboard layout which is the same as the US layout, but with a change such as putting the euro symbol (€) at a more convenient location, the best starting point would be the US layout.

If you are creating a keyboard layout for a language or script which doesn't yet have a standard, you may still be able to find a base keyboard to use. If not, though, it may be simpler to start from a completely blank keyboard layout. This allows you the most freedom, though it requires the most work. Avoiding some of this work is possible with some other tools, such as KeyLayoutMaker (http://scripts.sil.org/keylayoutmaker).

6.3. Creating a keyboard layout in Ukelele

Once you have done the design work, and know what the keyboard layout will look like, and have decided on a starting point, you are ready to actually create the keyboard layout.

As possible starting points, all the system keyboard layouts as of Mac OS X 10.4.11 (Tiger) are available on the disk image on which Ukelele comes. There are also other keyboard layouts on the installation disk image, including Logitech keyboard layouts. Other keyboard layouts are available on the internet. See the Chapter 10, Resources, at the end of this manual for some pointers to available keyboard layouts.

6.3.1. Starting from an empty keyboard layout

An empty keyboard layout is very basic in what is defined. There are the usual definitions of output for the special keys (return, escape, delete, tab, space, arrow keys, function keys, help, page up, page down, home, end, etc), plus a basic set of modifier combinations (no modifiers, shift, option, caps lock, option plus shift). None of the letter, number or symbol keys have any output associated with them, so you truly have a blank keyboard layout, and you have to add all the output from scratch.

6.3.2. Starting from an existing keyboard layout

Most of the time, you can find a keyboard layout that is close to what you want, which means that you can start with that and make whatever modifications you need. In this case, open Ukelele, and choose File > Open (⌘O) to bring up a dialog box which allows you to choose an existing keyboard layout as your starting point.

If the starting point is an unbundled keyboard layout, select File > Duplicate (⇧⌘S), and you will get a new copy created. If it is a keyboard layout in a keyboard layout collection, select

the keyboard layout in the list, and select File > Duplicate Keyboard Layout, and it will create a new, untitled keyboard layout for you to save where you wish.

Where might you get such keyboard layouts? Apple supplied several different keyboard layouts. Most of these (including the Unicode keyboard layouts) have been converted to a form that Ukelele can edit, and are in the folder "System Keyboards" on the disk image that Ukelele is on. You will notice that these are divided into five collections, Central European, Cyrillic, Dvorak, Roman and Unicode. Please note that these keyboard layouts only produce characters in the corresponding script (Dvorak produces Roman script). If you want to make these into full Unicode keyboard layouts, you will need to change the keyboard ID and script. See further discussion of scripts earlier. Of course, those keyboards in the Unicode folder are already set to produce Unicode.

Other keyboard layouts are available on the internet. In chapter 10, you will find a selection of examples. Please be sure to check the conditions of use for each of these. Some are commercial, others are shareware, and others are freely available. If you find other sites that are helpful and not listed, please let us know, and they can be added to the list.

6.3.3. Starting from a non-XML keyboard layout

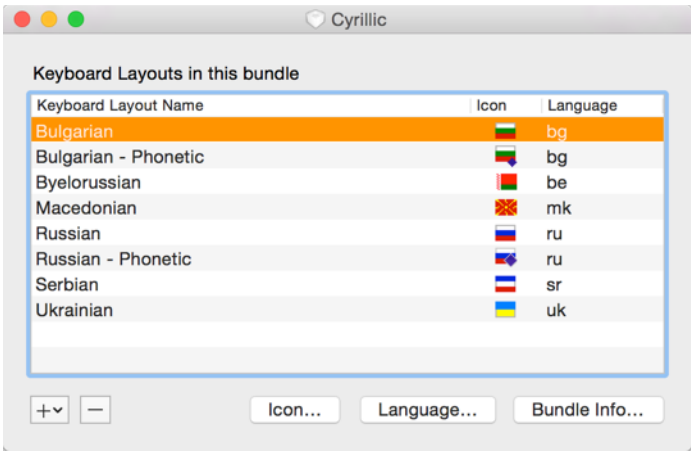
Sometimes you have a keyboard layout that is similar to what you want to use, but it is not a type of keyboard layout file that Ukelele can edit. This would include old resource-based keyboard layouts (uchr), but also other keyboard layouts supplied with the operating system. Since Mac OS X 10.5 (Leopard), the system keyboard layouts are not XML files, and so cannot be directly opened in Ukelele. Mac OS X 10.7 (Lion) brought several new keyboard layouts, and these can only be accessed through Ukelele.

Whatever the source, if you can select the keyboard layout as the current input source from the input menu, then you can convert it to a form that Ukelele can edit. This is done by selecting File > New From Current Input Source. This will take the current keyboard input source and convert it to an XML file which is then opened by Ukelele. Alternatively, you can add the current input to an open bundle by clicking the + button to the lower left and selecting Capture Current Input Source.

If you have a very old KCHR resource-based keyboard layout, you will not be able to convert it using Ukelele 3.0 or later.

6.4. Working in the Ukelele windows

There are two main windows in Ukelele. One is where you can see a whole collection, with a list of all the keyboard layouts, plus their icons (if any) and associated languages (if any). The other is a representation of a keyboard, and is where you do the work of changing output of keys, defining modifier sets, creating dead keys, and so on.



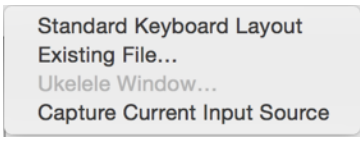
6.4.1. The collection window

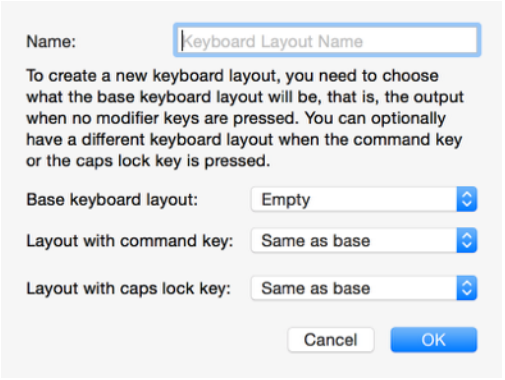
The keyboard layout collection window looks like the figure. It consists of a table listing all the keyboard layouts contained in the collection. There are also various buttons to make changes to the collection or individual keyboard layouts

within the collection.

For each keyboard layout, the name of the keyboard layout is listed, as well as its icon, if any, and language, if any. Double-clicking a keyboard layout opens a keyboard window for that keyboard layout. Choosing "Open Keyboard Layout" from the File menu will do the same.

The + button opens a menu with different options for adding a keyboard layout to the collection. Choosing "Standard Keyboard Layout" opens another dialog allowing you to create a keyboard layout based on one of a number of standard layouts (QWERTY, QWERTZ, AZERTY, Dvorak or Colemak), optionally with different keyboard layouts with the command or caps lock keys.





Choosing "Existing File..." brings up a standard open file sheet, which allows you to choose a keyboard layout file (not a bundle with a collection), and add that keyboard layout to the collection.

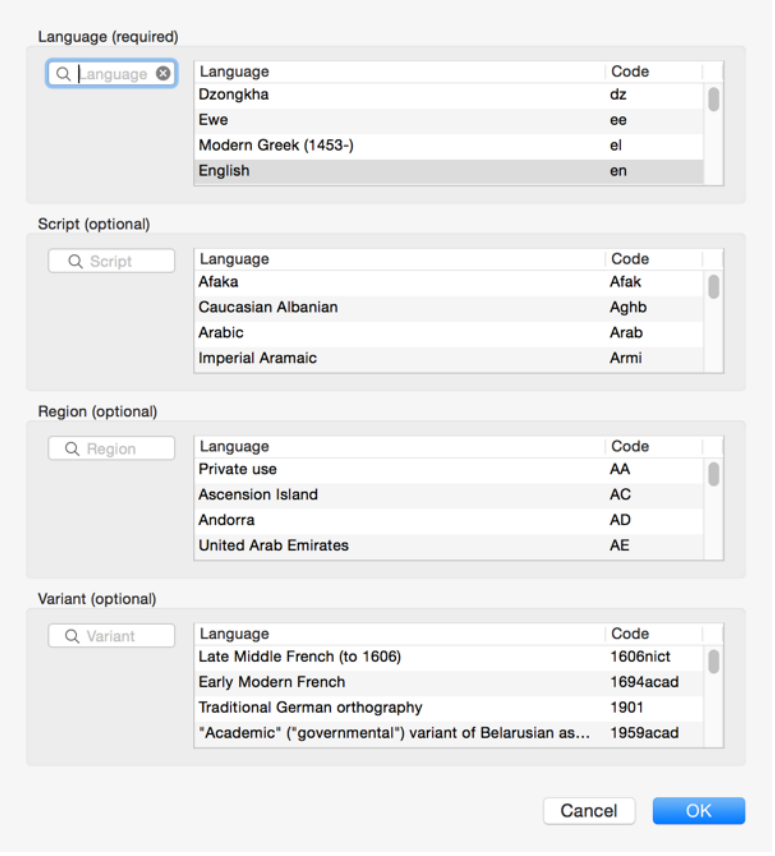
If you have a keyboard layout open to its keyboard window, and it is not part of the current collection, then "Ukelele Window..." will be available, and this will allow you to copy that keyboard layout into your collection.

The last option, Capture Current Input Source, reads the current keyboard input source and converts it, if possible, to an XML representation, and adds it to the collection. If an icon and/or language are specified for the current keyboard input source, they will be set in the collection, too. This is often the most convenient way to start a new keyboard layout based on one of Apple's more recent keyboard layouts, which aren't currently available as XML versions.

The - button removes the selected keyboard layout from the collection.

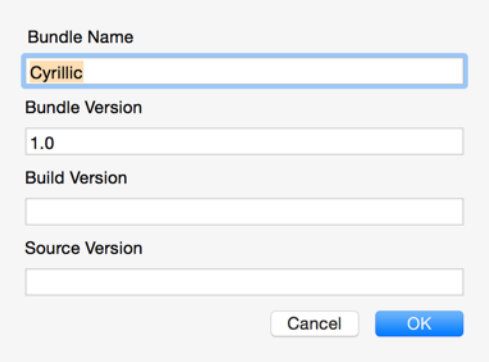
The Icon... button brings up a standard file open dialog which allows you to add or change an icon to the selected keyboard layout. You can also add an icon via drag and drop.

The Language... button allows you to set the intended language for the keyboard layout. Only certain languages are supported by Apple. The full list is given in Appendix A, but basically it contains most major languages, especially those of Europe and parts of Asia. Some also have different regions associated, but these largely affect things like currency symbols and the like.



Languages use the BCP 47 standard, which allows language, script, region and variant to be supplied. The dialog that is shown when you click the Language... button or choose Keyboard > Set Keyboard Language... allows you to set values for all four, though only the language is required. At present, Apple

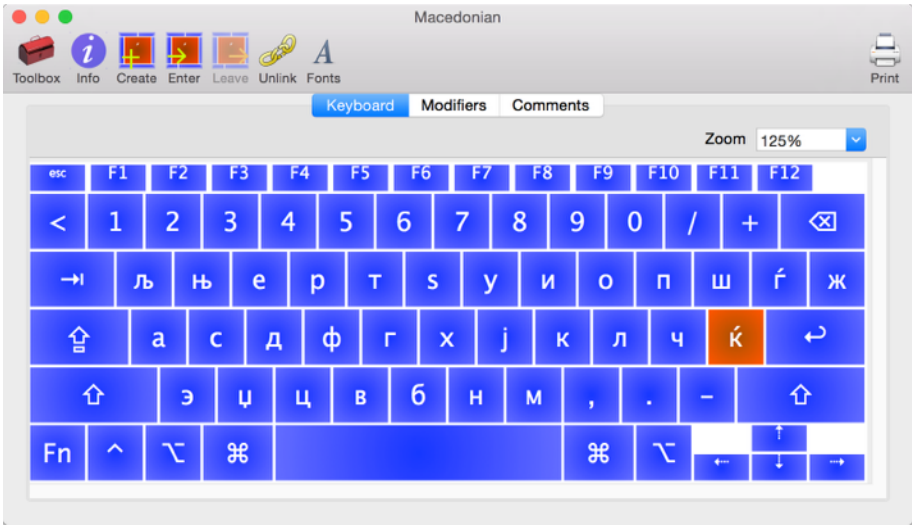
supports only two cases where specifying the script is useful, that of the Serbian language, which can use either Latin or Cyrillic script, and Chinese, which can be Traditional or Simplified. No variants are currently supported, and, in fact, those listed by Apple are non-standard, so there is little reason to specify a variant. See the appendix for a list of languages, regions and scripts supported by OS X.



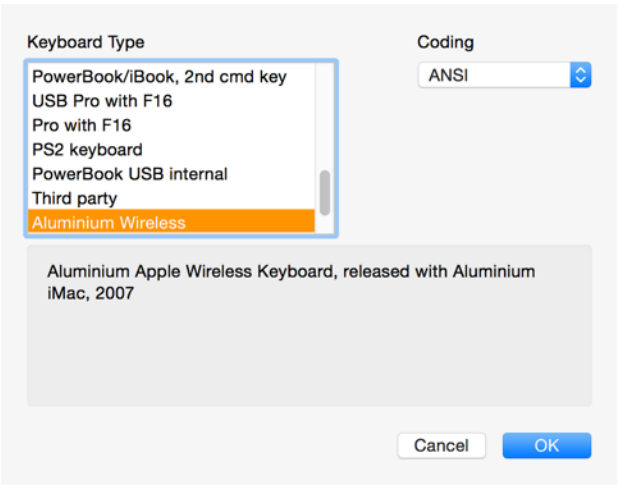
The Bundle Info... button allows you to set information about the bundle itself. This allows you to set the name of the bundle and some version numbers, which are mainly for the developer's benefit.

6.4.2. The keyboard window

When you create a new keyboard layout, or open an existing one, the main Ukelele window opens. This can also be called the keyboard window, as it shows a graphical representation of a keyboard. In most cases, the keyboard will look like the keyboard attached to your computer. However, if you have a non-Apple keyboard, or sometimes even if you haven't touched the keyboard since launching Ukelele, you will get a default keyboard. This is for display purposes only, and does not affect the use of your keyboard layout at all. A keyboard layout works with all hardware keyboards that are attached to the computer, and the display in Ukelele is only to show you how it looks on a range of different keyboard types.



You can change the display to a different keyboard type, so that you can see how various types of keyboard work, in their arrangement of the keys apart from the standard alphabetic and numeric keys. This is done by choosing "Keyboard Type..." from the View menu. However, before going into this, we need to look at another issue.



There are three standards for keyboards, ANSI (American National Standards Institute), ISO (International Standards Organisation) and JIS (Japanese Industrial Standard). These in general define what keys are on a keyboard, but not their exact layout. ANSI and ISO keyboards tend to be almost identical, with one extra key on an ISO keyboard, to the left of the Z key on a standard US keyboard layout. JIS keyboards have other keys for enabling input of Japanese

characters, but also they often rearrange some of the punctuation characters.

When you choose “Keyboard Type...”, you are presented with a dialog that allows you to choose what kind of keyboard the display should show. The list on the left has the classes of keyboard, and the popup menu on the right lists the coding standards (ANSI, ISO or JIS) available for the class chosen. A description of the keyboard is shown in the lower part of the dialog.

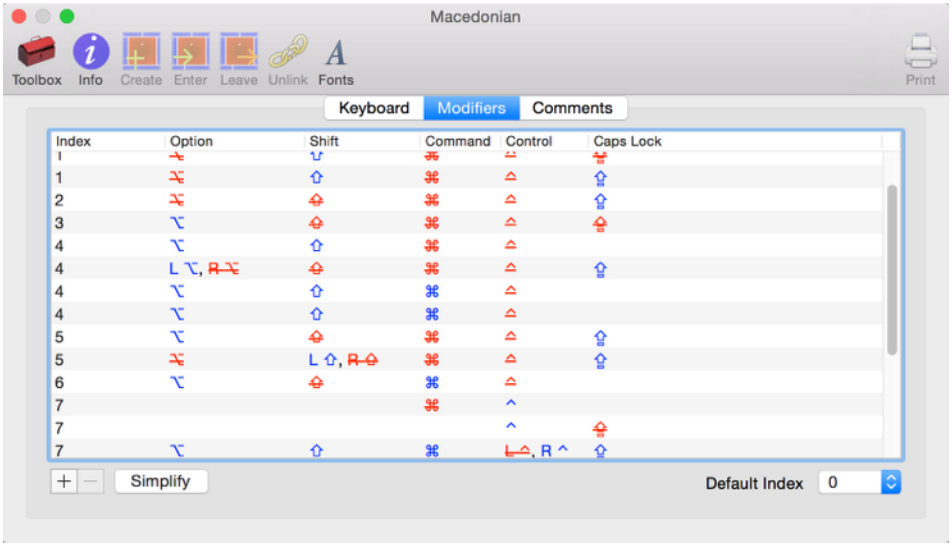
Once you have your window looking the way you want it, try pressing some keys and see what happens. When you press a modifier key, such as shift or option, you will see the appropriate key change to indicate that it is now down, and the rest of the keys will also change to show their output with that modifier. When you release the modifier key, the window will update to show the correct output for each key.

If you press a non-modifier key, the key that you pressed will show that it has been pressed by indicating that it is down. It will also toggle whether the key is selected. So, pressing a key will show it as selected and down until you release it, when it will be selected and up. Pressing again will make it unselected and down until you release it, when it will return to unselected and up.

If you choose a keyboard that is different to the actual keyboard you have attached to your computer, you can see the different arrangement of the keys coming into play here. You may discover that there are some keys that you don’t have on your keyboard, or that some keys on your keyboard don’t correspond to anything on the keyboard shown in the window. This is useful, both for seeing how different keyboards may end up implementing your layout, and for getting access to some keys you may not have on your keyboard, such as if you have a laptop.

There is an option in the View menu called “Sticky Modifiers”. This is a convenience for helping you to get modifier combinations without having to hold down all the modifier keys. In a normal keyboard, the caps lock key functions like a toggle switch: one press makes it active, a second press makes it inactive. Usually, there’s a light or other feedback to tell you that caps lock is active (or down). Sticky Modifiers makes all the modifier keys behave the same way within Ukelele. Press the shift, option, command or control key once, and you will see it stay down in the Ukelele window. Also, you can click the modifier keys in the Ukelele window, and they will change state, just as if you’d pressed the corresponding key.

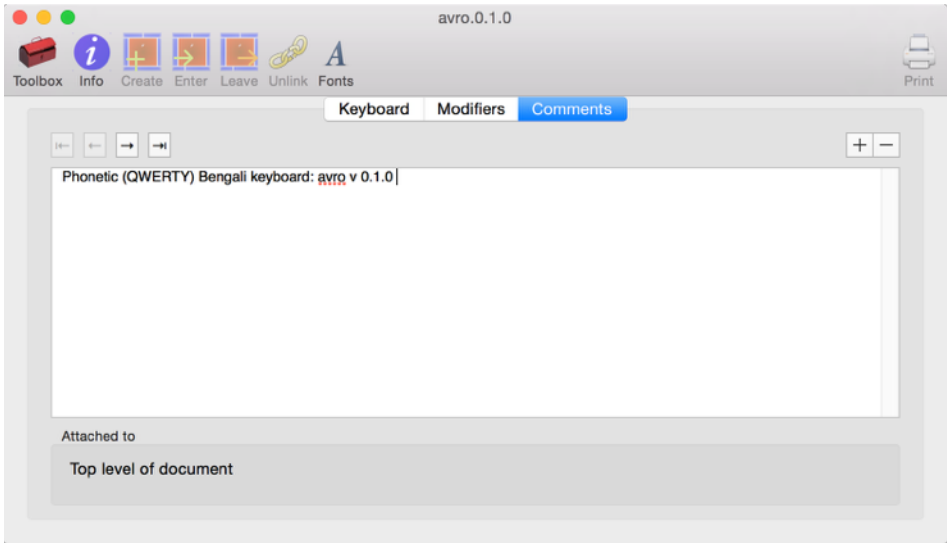
You’ll notice that the keyboard window has a tabbed interface, with the tabs being “Keyboard”, “Modifiers” and “Comments”. What we have looked at so far is the Keyboard tab.



The Modifiers tab allows you to create and edit modifier sets. With five different modifiers (shift, option, command, caps lock and control) and two different states for each, there are 32 potential sets of modifiers. Very few, if any, keyboard layouts will use anything like that number, and usually somewhere in the range of 5–12. Modifier sets allow you to group the modifier combinations as you wish.

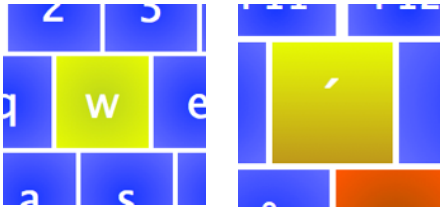
You will find much more about modifier sets, including how to specify them, create and modify them, and simplify them, in section 6.8 below. All the features of the modifiers tab are explained in that section.

The Comments tab allows you to see the comments that are inserted into the XML file which defines the keyboard layout. They are useful for such things as copyright notices, contact information and notes on decisions made in designing the keyboard layout. There are also automatic comments inserted by Ukelele when creating a new keyboard layout, and a time stamp for when it was last edited (unless that option is turned off in Ukelele's preferences).



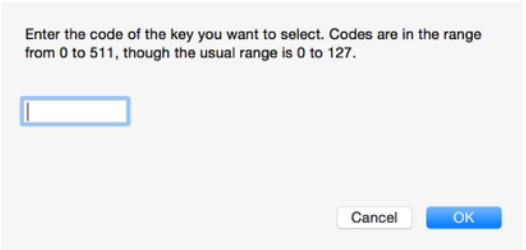
6.5. Selecting keys

An important part of many operations in Ukelele is selecting the key which will be changed by the next command, whether that be a menu command, a toolbar button, a contextual menu command, or a key equivalent of a menu command.



A selected key shows in a different colour, which varies depending on whether the key is up or down, and an ordinary key or a dead key. Shown are a selected ordinary key and a selected dead key.

There are three ways to select a key. The first is to type the key with your hardware keyboard. Selection is a toggle, so that one press selects the key, and a second deselects it. The second way is to click the key on the screen. Again, this is a toggle, so a second click deselects the key.



The third way is to choose Keyboard > Select Key By Code..., which brings up a dialog to allow you to enter a key code and select the key with that code. This is also a toggle, so entering the key code of the key that is already selected will deselect that key.

6.6. Editing key output

The most common operation in editing a keyboard layout is editing the output of a key. You want to make the keys produce the characters you plan for them, and this is how it is done. There are two basic ways of doing this: drag and drop, and double-clicking a key in the window.

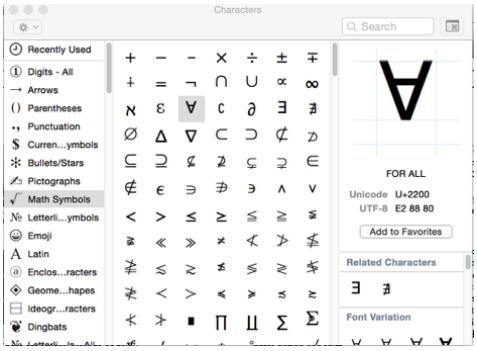
6.6.1. Drag and drop

When you have a fairly simple set of keys to work on, drag and drop is often the easiest way to do the editing. The method is extremely simple: drag the character or characters that you want onto the key in the Ukelele window, and that sets that key to produce the character or characters that you dragged.

OK, so where do you get the characters you want? The obvious place is the Character Viewer. You will find it in the Edit menu, the item Emoji & Symbols. It is also available in the input menu, assuming that you have the input menu enabled in System Preferences in the Keyboard (or Language & Text) pane.

If you haven't got the Character Viewer available in Mavericks or earlier, you need to go to the same place in System Preferences, except that it is the Keyboard tab (in Mavericks), where there is a check box "Show Keyboard and Character Viewers in menu bar". On Mountain Lion, you need to first find the Character Viewer (or Palette) in the list on the Input Sources tab, and ensure that the check box to the left is checked. Once it is, the input menu will now have an item "Show Character Viewer" or "Show Character Palette".

The Character Viewer is a floating window which offers access to every character in Unicode. Its appearance is different in every major version of OS X, as Apple struggles with good ways to present a huge amount of data in a way that people find helpful. All versions offer various groupings of characters, organised in different ways, such as by script (Roman, Cyrillic, Japanese, etc), by category (punctuation, symbols, etc), or by code point (the numerical code for each character). Screen shots here show the 10.10 (Yosemite) version.



You use the Character Viewer with drag and drop by clicking on its title bar (to activate the window), finding the character you want, clicking on it, and dragging it onto the Ukelele window, and dropping it on the key that you want to change. If you want to change the output for a key with modifiers down, such as shift-option, hold those modifier keys down before dropping the character onto the key. Note that you can also use "sticky modifiers", which were introduced earlier.

You don't have to use the Character Viewer, though. You can select text in a Unicode-aware application and drag that onto the key that you want to change. So, for example, if someone has put together a document which has all the characters you want, just open it in a program like TextEdit, Nisus Writer (Express or Pro), Mellel, AbiWord, Word (2004 or later), and you can then drag characters out of that program into the Ukelele window. Preview allows you to drag characters out, but Adobe Reader apparently does not.

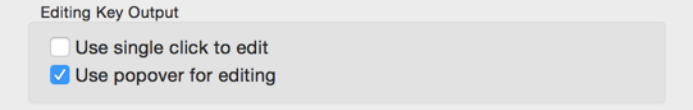
Note also that you can use a Ukelele document as your source. You can drag and drop the output of keys (apart from the special keys), either within a window or across two Ukelele windows. So, if you have a keyboard layout you want to use as a basis, you could have the original and your version open, and drag characters from the original to the new positions in your keyboard layout.

One reason for using a document rather than the Character Viewer is if you want to have a key generate more than a single Unicode character. An example might be where you want to use a combining diacritic with another character, or where you want to produce a digraph of some sort. Some people even make a key that produces some short text, such as their name, or email address, as long as it is less than 20 characters.

There are also other alternatives for access to Unicode characters, such as PopChar http://www.ergonis.com/products/popcharx/) or Unicode Checker (http://earthlingsoft.net/UnicodeChecker/). For Chinese script, Unihan Variant Dictionary (http://www.ideographer.com/unihan/) may be useful.

6.6.2. Double-click

When you don't want to use the Character Viewer, or want to use some characters that are easier entered as code points, or if you want to provide more than one character as key output, or you just don't like drag and drop, the alternative is to double-click the key in the Ukelele window. This brings up a dialog which allows you to enter the output you want. In the Ukelele preferences, there is an option to allow you to use a single click on a key rather than a double click to open the dialog. You can also choose whether you have a popover (the default) or a sheet for the editing dialog.



As with drag and drop, you should hold down the modifier keys that are appropriate for what you want to change. So, if you want to change the output for a key with caps lock and option down, have caps lock enabled and the option key held down when you double-click the key. Alternatively, use "sticky modifiers" to make all the modifier keys behave like caps lock, with each press changing state from up to down and vice versa. Sticky modifiers also allows you to click a modifier key to change it from up to down and vice versa. With sticky modifiers, you set the modifiers the way you want them, and then double-click the key.

The dialog allows you to enter text just about any way you want, including using the Character Viewer, by double-clicking, using the Insert button (if it exists) or drag and drop. This is another way of allowing strings of more than one character to be assigned to a key.

You can use the dialog to specify a character by its code point. Every Unicode character has a unique number, which is usually expressed as a hexadecimal (base 16) number, but can also be a decimal (base 10) number. For example, the inverted exclamation mark (¡) has the code point U+00A1, which means that the numerical code is A1 (hexadecimal) or 161 (decimal). For characters in the Private Use Area (U+E000 to U+F8FF), this can be the only unambiguous way of describing a character, since such characters vary from font to font.

To specify a character by its Unicode code point, you have to use the special XML notation for encoding a value. It is very important that this be entered exactly, or the XML parser will not be able to recognise it. This notation comes in two forms, depending on whether you want to use hexadecimal or decimal notation. To enter U+00A1 in hexadecimal, you use "¡" without the quotation marks. To enter it in decimal, you use "¡" without the quotation marks. The difference is the "x" after the hash (#) mark, which marks the number as hexadecimal. There is a bit of flexibility: case is not important within the number, and leading zeroes can be omitted, so that "¡" is just as valid. However, note that it *must* be a lowercase "x": using "X" does not work.

If you want to provide multiple characters via their Unicode code points, just run them together, with no intervening spaces, such as "𝄐𝄫" specifies "𝄐4ン4ン+". If you

put a space, it is considered part of the output, so "𝄐 𝄫" includes the space, so it is "リ4ン リ4ン+"

Note that there is a maximum of twenty Unicode characters that can be produced by a single key. If you try to make it more than twenty, Ukelele will tell you that it cannot do this.

6.7. Creating and editing dead keys

Dead keys are a powerful concept, and with power comes complexity. Some aspects are pretty straightforward, but sometimes you may wish to create a more complex keyboard layout with more advanced dead key use. Hopefully, the discussion in this section will progress from the more basic to the more advanced, and you won't have to get into all the details that you don't need.

6.7.1. What use is a dead key?

The basic reason for having a dead key is to allow modification of characters. The most familiar example for those who use Roman script keyboard layouts is adding accents. On the standard US keyboard layouts, typing ⌥u followed by a vowel gives you the vowel with two dots (diaeresis, umlaut), so that typing ⌥u u gives ü, ⌥u a gives ä, and so on. In this case, the dead key is ⌥u.

A dead key produces no characters on its own. It is only in combination with the next key typed that it produces output. In this way, it is best thought of as a way of modifying output. It is possible to do other things with dead keys, but that is the main idea.

6.7.2. Principles for designing dead keys

The most important factor in designing a dead key is that the user be able to remember how to use it! Often, there are only a small number of combinations that are obvious, such as those that are established conventions in the Mac world, such as ⌥e for an acute accent, or ⌥n for a tilde. That means that you will often have to establish your own conventions for your purposes.

Dead Key States

A dead key state is a state in the finite state automaton (or machine) that the keyboard layout implements. What that means is that entering a dead key state changes the way that the next key typed behaves. There is a special state called "none", which is the default state, where the system begins. Other dead key states have arbitrary names. Ukelele can create names for you, which (with default values) are called "Dead Key State n", where "n" is a positive integer. Otherwise you can create your own names.

Dead key state names are never seen by the user, and are only used when editing the keyboard layout. Often, you don't care what name a dead key state has, so you might just as well let Ukelele set the name for you. However, there are times when you do need to have a name that you can remember. When you want to edit the dead key later, or when you want to use it as part of a multi- level dead key, it is good to know the name.

A good name for a dead key state is one that is memorable, and usually is chosen to reflect the intent or function of the dead key. For example, on the US keyboard layout, the dead key state entered by typing ⌥-e is called "acute", as it generates letters with acute accents.

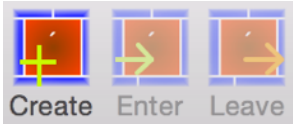
Consistency is very important. If a dead key produces a particular modification, such as adding a diacritical mark to a letter, then it should do the same for other letters, when this makes sense. A dead key that produces one diacritic for some letters, and a different one for other letters, will be confusing for the user. In that sort of case, two different dead keys would be preferable.

Another consideration is that the most frequently typed characters should be easy to type. In other words, dead keys are not always the best solution. For example, if you were creating a keyboard for Finnish, which uses ä and ö extensively, then a dead key would slow down a typist compared to a keyboard layout which had single keystrokes for those letters.

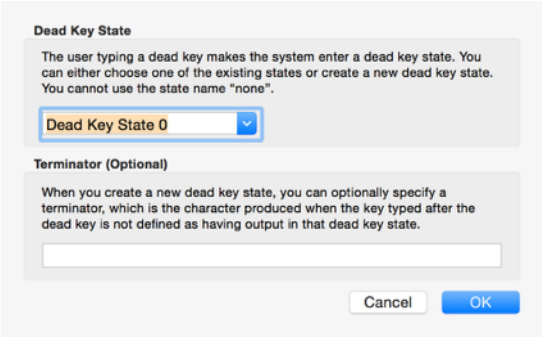
To say it again, plan carefully how the keyboard layout will work before you start to create it in Ukelele. Dead keys are an important part of the design. If they aren't easy to use, they won't be used, and the work you put into it hasn't turned out to be helpful to the user.

6.7.3. Creating a new dead key

Once you have decided on a dead key, you create it by first selecting the key, and then choosing "Edit Dead Key..." from the Keyboard menu, or click on the "Create" button in the toolbar. Whichever of those two options you use, the modifiers used are whatever are currently down when you click the button or choose the menu item, or those that are shown down when you have "sticky modifiers" turned on.



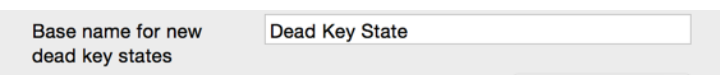
The next thing that happens is that you are presented with a dialog asking you for the name of the dead key state and for the terminator. The dead key state is the state that your new dead key will enter. For more information about dead key states, see the sidebar "Dead Key States". The terminator is the character the dead key produces when nothing is defined for the next key. See below for more on terminators.



There are two options for the dead key state, either creating a new dead key state or an existing state. The dialog begins with a default name generated by Ukelele. The name that is generated as a default is set in the preferences, and can be any string, to which a number is appended to create a unique name for the dead key state.

You have the option to create a new dead key state name, which could be either the default name generated by Ukelele, or whatever you want to call it. Otherwise, you can choose an existing dead key state from the list.

The name of the dead key state is never shown to the user, so it's perfectly OK to use the name automatically generated by Ukelele. The name of the state is used when editing the keyboard layout, so it can be worthwhile giving it a name that is memorable to you. If you don't like the name that you've given the dead key state, you can change the name later by choosing "Change State Name..." from the Keyboard menu.



If you give an invalid name for a dead key state, whether that be blank, or "none" or "0", then Ukelele will tell you this, and ask you to correct it.

The terminator for the dead key state is the output when the next key typed has no output defined for the dead key. That sounds confusing, but it's a fairly straightforward concept. For

example, in the US keyboard layout, ⌥e plus a vowel produces an acute accent over the vowel, so that ⌥e followed by a produces á. If you type ⌥e and then a character that doesn't take an acute accent, such as a digit, you get the terminator, which is an acute accent on its own, plus the digit. So, typing ⌥e 4 gives you ´4. See section 6.7.8 on terminators for more information.

Once you have created your dead key, assuming that it is a new dead key state, most of the keys will appear greyed out. What you see is the output of each key in the new dead key state. A normal-looking key will have output in the dead key state, while the greyed-out keys show the output in state "none", for your reference.

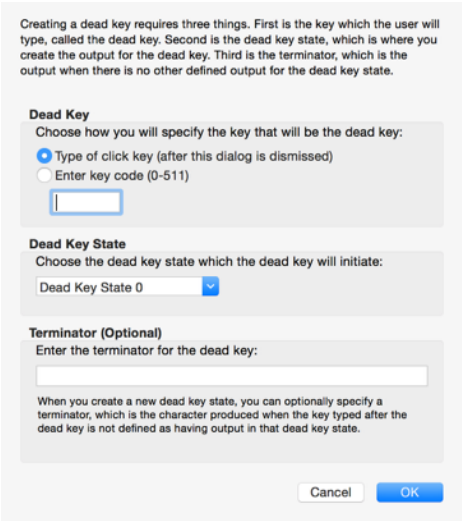
You can verify that you have entered a dead key state by looking at the current state section in the inspector. This is a floating window that shows the current stack of dead key states. The current state is at the top, and the last entry will always be "none", the state when there are no dead keys active.

In the dead key state, you edit output in exactly the same way that you do for normal keys. Drag and drop output, or double-click keys to edit the output in a dialog.

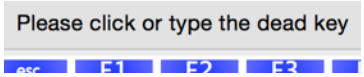
When you are done, you can exit the dead key state by choosing "Leave Dead Key State" from the Keyboard menu, or click the "Leave" button on the toolbar. That takes you back to the state you were in before you created the dead key.

6.7.4. Creating a dead key without a selected key

If you have not selected a key before you select Keyboard > Create Dead Key... or clicked the Create button in the toolbar, then you are presented with a different dialog. This has the same options as the case where you have a selected key, but there is an extra section at the top of the dialog, where you specify the key. There are two options for this, either typing or clicking the key after the dialog is dismissed, or by entering the key code.

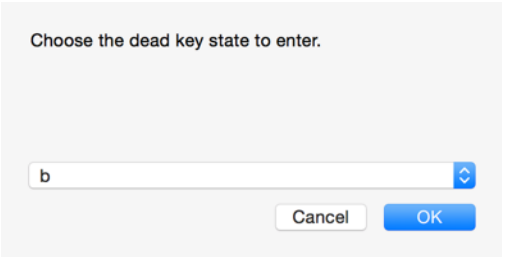


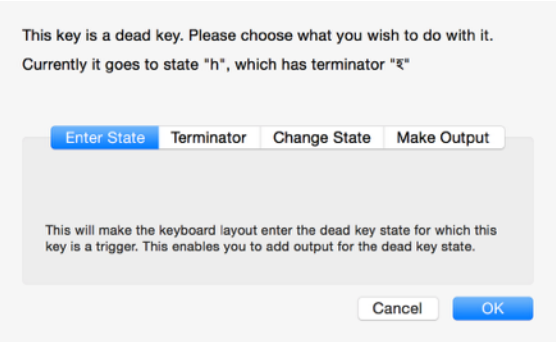
If you don't give a key code, then Ukelele prompts you to type or click the dead key. Doing so completes the process of creating the dead key.



6.7.5. Editing an existing dead key

There are times when you want to modify an existing dead key. For example, you may want to add another accented character to a keyboard layout that already has a dead key that produces that accent. Or, you may have made an error with a dead key, and you want to change it. Another case is when you want to create a multi-level dead key (see section 6.7.7 for more on this).



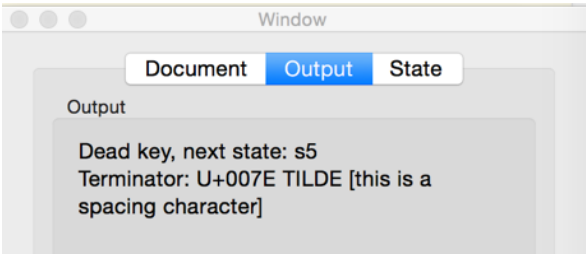


The procedure for editing an existing dead key is much the same as for creating a new dead key. Select “Enter Dead Key State...” from the Keyboard menu, or click the “Enter” button on the toolbar. This time, choose the name of the state that you want to edit from the pop-up button in the dialog, and click OK.

What if you don’t remember the name of the dead key state? Simply double-click the dead key in the Ukelele window, when it is shown with the dead key colour (red in the default colour theme). You will see a dialog that asks what you want to do. Choose the Enter State tab and click OK to edit the dead key state.

Alternatively, select the dead key, then choose “Create Dead Key...” from the Keyboard menu, or click the “Create” button on the toolbar, and type or click the dead key. This will bring up the same dialog.

6.7.6. Moving a dead key



Sometimes, you want to move a dead key. For example, if you are starting from an existing keyboard layout, you may not want the dead keys where they are, but want to use a different key for the dead key. As a concrete example, suppose you started from the Spanish keyboard layout, but you want to use a different dead key for the tilde combinations, say ⌥j instead of ⌥ñ.

The procedure is pretty simple in concept. What you want to do is to make a new dead key that behaves exactly like an existing dead key. Doing that takes a few steps, but they are fairly straightforward.

The first thing that you need to do is to work out the name of the dead key state. The easiest way to do this is to use the inspector. From the View menu, select Show Info, or click the “Info” button on the toolbar. This brings up a floating window. When the mouse is over a key, it shows the output for that key as Unicode code points. More importantly for this purpose, it shows the name of the dead key state when the mouse is over a dead key.

Move the mouse over the dead key, with whatever modifier keys are necessary. So, in the Spanish example, hold down option (⌥), and move the mouse over the key that produces ñ without the option key. Remember the name of the dead key state once you have it. In the Spanish example, it turns out that the dead key state name for the tilde is “s5”.

Next, you should press your new dead key (such as j in the example above) to select it, and then hold the option key down (or use sticky modifiers and press it once), then choose “Create Dead Key...” from the Keyboard menu, or click the “Create” button on the toolbar. When the dialog box appears to ask for a name for the dead key state, choose the state (s5 in the example) from the pop-up menu, and click OK. You should then see that you are in the correct dead key state. Choose “Leave Dead Key State” or click the “Leave” button on the toolbar to return to state “none”.

Finally, and optionally, you can delete the original dead key. For instructions on that, see section 6.7.10 below.

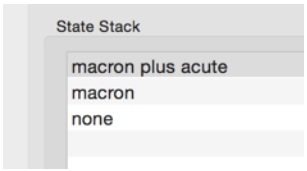
6.7.7. Multi-level dead keys

There are occasions where you want to create multi-level dead keys. These are combinations where you type one dead key to enter a dead key state, and then type a second dead key to enter another dead key state. There is no practical limit to the number of levels, though it would be rare to go beyond two or three.

An example would be when you are creating a keyboard layout that is used for a language where you need to specify both length, by means of a macron over a vowel, and tone by means of an acute accent over the vowel. With only single dead keys, you would need to have separate dead keys for macron, acute accent, and both macron and acute accent. With multi-level dead keys, you need only two dead keys, but create three dead key states.

Continuing with the example, we could have ⌥a as the dead key for a macron, and ⌥e for an acute accent. We first create the ⌥a dead key. First, ensure that we are in state "none" by checking the current state in the inspector (or checking that the "Leave Dead Key State" menu item or "Leave" button in the toolbar are disabled). First, hold down ⌥a, then choose Edit Dead Key... from the Keyboard menu or click the "Create" button on the toolbar. In the dialog, enter "macron" as the name of the dead key state (replacing the default new name). For the terminator, you can leave it blank, or enter "¯", the macron character, U+00AF. You should now see that the inspector shows a stack of two states, macron and none. Now proceed to add output in the macron state, which should be at least the vowels with the macron.

Next, we create the second dead key state, by typing the dead key for the acute accent ⌥e, then choosing Edit Dead Key... again or clicking the "Create" button on the toolbar. Name this dead key state "macron plus acute". There is no single Unicode character which is macron plus acute, so the terminator will need to be the macron (U+00AF) plus the combining acute accent (U+0301) or the acute accent (U+00B4) plus the combining macron (U+0304). You will see that the current state palette shows three states: macron plus acute, macron, and none. Add the output you want for this state.



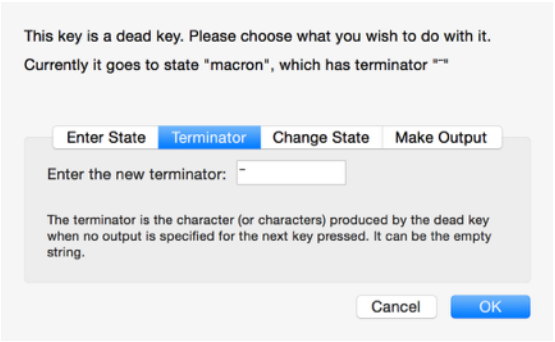
Now, we need to leave both dead key states. Choose Leave Dead Key State from the Keyboard menu or click the "Leave" button on the toolbar, and you will see the current state palette show just macron and none. Choose Leave Dead Key State again, and you will be back to state "none".

The next step is to create the dead key state for the acute accent. Again, type the dead key, ⌥e, then choose Create Dead Key... or click the "Create" button on the toolbar. Make the name of the dead key state "acute", and the terminator (if you want it) the acute accent, U+00B4. Then add the output for the acute accent state.

To complete the work, type the dead key for macron, ⌥a, then choose Create Dead Key... again or click the "Create" button on the toolbar. When the dialog comes up, select the name we had before, macron plus acute. You will see that the current state section of the inspector shows three states: macron plus acute, acute and none, and the keyboard shows the state you created above. Choose Leave Dead Key State or click the "Leave" button on the toolbar twice, and we are done.

6.7.8. Terminators

The terminator of a dead key state is the output produced when the key typed after the dead key produces no output in the dead key state. To see how this works, consider how you handle creating a dead key. You create a new dead key state, and then add output for various keys. For those that you don't add output, there is no output defined for that dead key state. What



happens when you type the dead key, and then one of those keys that doesn't have output defined? The answer is that you get the terminator of the dead key state, then the output for the key you typed in the normal state.

Terminators can be created when you create a dead key. Otherwise, they can be changed at any time. First, enter the dead key state, if you are not in that state already, by choosing Enter Dead Key State... from the Keyboard menu or clicking the "Enter" button on the toolbar, and choosing the dead key state from the pop-up menu in the dialog. Then choose Change Terminator... from the Keyboard menu. In the dialog that appears, the old terminator is in the text field, and you can replace it with the new terminator.

An alternative way to change the terminator of a dead key state is to drag output onto the dead key. This has to be done when the you are in a state other than the dead key state, that is, in a state where you see the dead key shown as a dead key, showing the terminator. Ukelele shows a dialog asking what you mean to do, and changing the terminator is the default option.

A third option is to right-click (or control-click) the dead key, and choose "Change Terminator" from the contextual menu.

A final option is to double-click the dead key, then choose the Terminator tab, and enter the new terminator in the text field.

6.7.9. Importing dead keys

If you are creating a set of keyboard layouts, or creating a keyboard similar to an existing one, you might want to use a dead key from one keyboard layout in another. This can be done in Ukelele, by using the Import Dead Key... command.

Importing a dead key means copying a dead key state, meaning all the output associated with that state. You can choose the key that will become the dead key in the keyboard that you are editing, as well as the name of the dead key state.

There is one condition that must be satisfied with the two keyboard layouts before a dead key can be imported. Both must have exactly the same set of modifier key combinations. That is, each keyboard must have the same number of different modifier key combinations, and each set must use the same set of modifier keys.

When you choose Import Dead Key..., Ukelele first asks you to choose the source keyboard layout, that is the keyboard layout with the dead key that you want to import. It does this by presenting you with an open file dialog, and you should open the keyboard layout in that dialog. Ukelele then checks to see that the modifier key combinations are the same. If not, you will get a warning dialog, and the process is cancelled.

Once you have opened the source keyboard layout, you are then presented with another dialog, which asks you to choose the dead key state to import. You choose the name from the pop-up menu. The dialog also asks you to name the dead key state in the keyboard layout you are editing. As usual, if the name is already taken, you will told so, and you can enter a new name.

At this stage, the dead key state has been imported, but as yet there is no dead key that will take you to that dead key state. To do that, you must create a new dead key, and select the imported state as the dead key state to enter.

6.7.10. Deleting a dead key

Deleting a dead key is quite simple. All you need do is give the dead key some new output, either by drag and drop or double-clicking the key (single clicking works if you set that option in the preferences). In either case, Ukelele will give you some options in a dialog. If you drag text to the key, then option is either to have that be the new terminator, or to convert the dead key to an output key. If you double click, you have the choice of entering the dead key state, entering a new terminator, changing the target dead key state, or making the key an output key and specifying the new output.

6.8. Managing modifier key combinations

You may not have thought a whole lot about modifier key combinations, since they just work in most keyboard layouts without too much thought. You press shift, and you usually get capital (or upper case) letters. Press option, and you get less common characters, or dead keys. Press caps lock, and you get capital letters, but the digits rather than symbols. And so on, through a variety of combinations. Even so, you need to think about them when you are designing and building a keyboard layout.

Any modifier combination maps a key to a particular set of outputs and dead keys. The essence of this section is to give you guidance on how to create the right sets and the right combinations of modifier keys that access these sets.

6.8.1. What is a modifier key?

Modifier keys are shift (⇧), option (⌥), command (⌘), caps lock (⇪) and control (^). On some keyboards, particularly portables (iBook, PowerBook, MacBook, MacBook Pro, MacBook Air), there is a key labelled “fn”, which appears to be a modifier key. However, it is not a modifier key in the sense that the other keys are modifier keys. The fn key only works to activate the keys that are otherwise unavailable on portable keyboards, such as the numeric keypad (if any), and navigation keys like page up, page down, home and end. In other words, the fn key converts one key to another, but does not act as a modifier key. The same is true of the num lock key, too, which is basically the same as the fn key, but sticky like the caps lock key.

The main point to notice is that the output for each key is specified with regards to the current modifier key combination. So, each combination of modifier keys can produce different output for a given key. A keyboard layout defines certain modifier key combinations as belonging together, so that, for example, shift down, caps lock up and shift down, caps lock down are often grouped together as being the same. The number of different modifier key combinations that a keyboard layout defines is the way that the number of different outputs for each key is defined.

There is no necessary correlation between two different modifier combinations, but that has to be done manually. In other words, what a key produces when the shift key is down is not automatically related to what the same key produces when the shift key is not down.

6.8.2. What is possible vs. what is useful

Apple defines modifier keys a little further, by adding right and left variations of shift, option and control. So, in theory, it is possible to have different combinations for the left shift key and the right shift key. If you look at the US Extended keyboard layout supplied in the System Keyboards folder with Ukelele, you will find that such modifier key combinations are defined, such as right shift and left control.

The problem with this is no current keyboards will actually supply different codes for the left and right modifier keys. In other words, the operating system will see both the left shift key and the right shift key as being the left shift key. The same holds true for option and control as well. Furthermore, any application that is built on Cocoa (which is the vast majority of Mac OS X apps) cannot see any right modifier keys as separate from left modifier keys.

The only keyboards that actually generate separate left and right modifier key combinations are very old, being ADB keyboards, which means that they cannot be used with Mac OS X 10.3 or later. No USB Apple keyboard (which is every current keyboard) will actually produce separate right and left modifier key codes. It is possible that a third-party keyboard could produce such codes, but that's not clear or predictable, and you probably wouldn't be able to use a keyboard layout that depended on that with any other keyboard.

So, the best advice is to just assume that you can only use the five different modifier keys in their various combinations. In theory, that gives you up to 32 different modifier key combinations, which should be sufficient.

A new feature in Ukelele 3.0 is simplifying modifier combinations. What this does is to remove any reference to right modifier keys, and convert any references to left modifier keys to being either modifier key. As long as you don't have a keyboard that can actually produce separate key codes and an application that can recognise them and make use of them, simplified modifiers are equivalent to the original set of modifiers. See further in section 6.8.3.1 below.

Unfortunately, not all those modifier key combinations are actually useful. Two restrictions are important. One is that using the command key as a modifier is dangerous, as you will want to use the command key for keyboard shortcuts, so the command key should only be used to generate basic key strokes. Have a look at the US Extended keyboard layout to see the standard way to handle the command key. Another way it can be handled is shown by the Dvorak-QWERTY keyboard layout, which maps the keys to their standard QWERTY arrangement when the command key is down.

The other restriction is the control key. In applications using the Cocoa text engine, which is most applications these days, control key combinations are intercepted at a low level and "key bindings" are applied. In practice, this means that what the keyboard layout specifies as output with the control key is not honoured by the system unless the option key is pressed. What happens is that the system evaluates the key stroke without the control key, and looks for a key binding of control plus that output. There are ways around some of these problems, but these are not practical for most people. For details, see the sidebar.

Alternatively, for most applications, you can use control key combinations as long as they also include the option key. So, you can have combinations like ⌥^, ⌥⇧^ or ⌥⇪^, and these should work as expected. This is generally preferable to modifying the system files to allow control key combinations on their own.

A third option is less useful, as it requires the user to remember how to do it, and requires two keystrokes for every control key combination. This is the use of the "quote" key binding. Normally, typing ^q means that the following key stroke will be handled without interpreting it as a command. So, if your keyboard layout has ^f set to produce ƒ (U+0192, Latin letter f with hook), typing ^q ^f should work. This depends on your keyboard producing a lowercase q when no modifiers are down, so it may not work if you have a non-Roman keyboard layout.

An average set of modifier key combinations would look something like this:

1. No modifier keys down.
2. Shift key down (often you have caps lock either up or down here).

3. Option key down.
4. Shift and option keys down.
5. Caps lock key down.
6. Command key down (often with the shift key either up or down).

Often, a separate combination for the control key down is added, though it's not essential.

Some suggestions about designing modifier key combinations:

- The most commonly typed characters should take the least keys, so it's not good to have a common character require two or three modifiers, unless one of them is caps lock, which you don't need to hold down.
- If you are providing a modification of a standard keyboard layout, like the Dvorak layout, you might consider making the command key use the standard layout, like the Dvorak-QWERTY keyboard layout does.
- One possible way to provide two different types of layout at once is to use the caps lock key to differentiate between them, so that one works with the caps lock key up, the other with the caps lock key down. For example, you might have Cyrillic with the caps lock key up, and Roman with the caps lock key down.

6.8.3. Creating new modifier key combinations

In most cases, when you begin a keyboard layout, there are certain modifier key combinations already defined. That may be all you need, but you can check it by looking at the modifiers tab of the keyboard window. You also edit the complete set of modifier key combinations from the modifiers tab.

6.8.3.1. Understanding the modifiers table

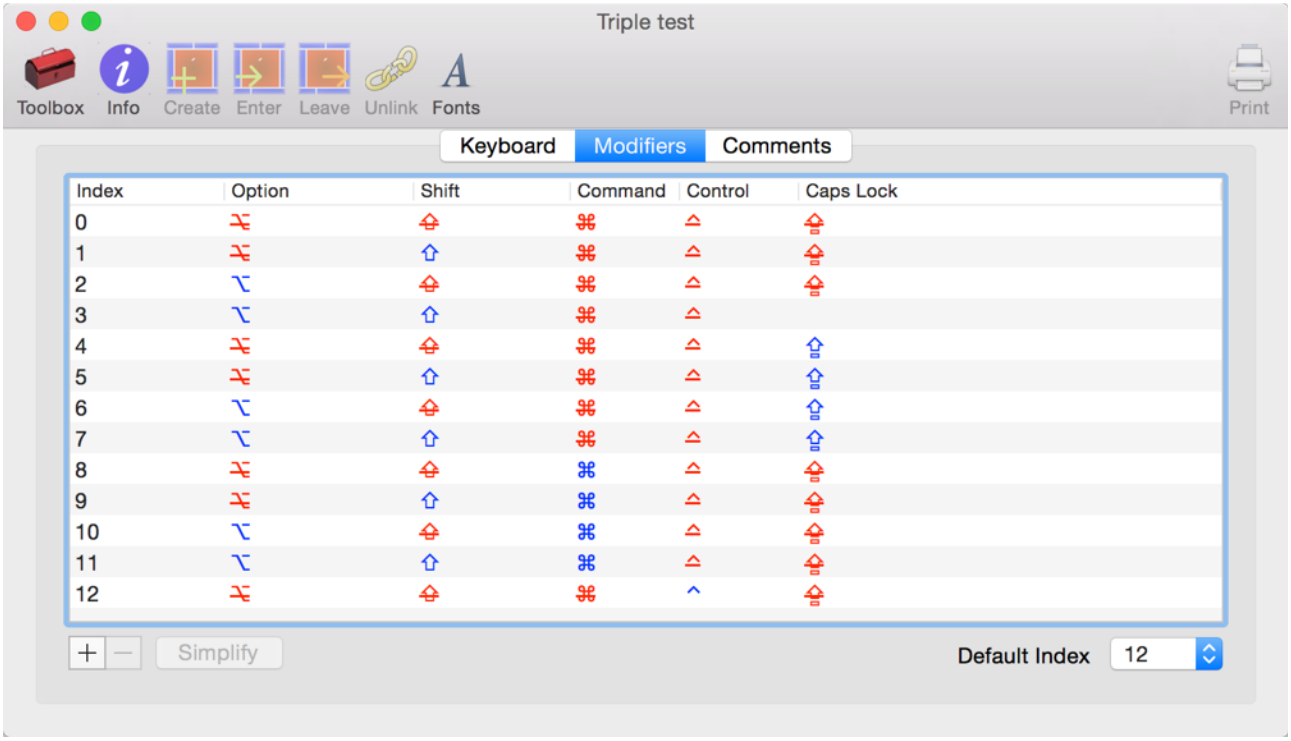
The modifiers tab can look a little intimidating at first. Part of the difficulty is the need to accommodate the left and right shift, option and control keys, which can be set independently.

For a particular modifier key combination, each of the modifier keys can be either down (pressed) or up (not pressed). To specify a modifier key combination, you say that each key either has to be one or the other (down or up), or that its state does not matter, so that it could be either down or up. For those modifier keys which have left and right keys, it is more complicated, in that there are more possibilities:

- Both up
- Either of them down
- Either of them either down or up
- Both down
- Left up, right down
- Left up, right either down or up
- Left down, right up
- Left down, right either down or up
- Left either down or up, right up
- Left either down or up, right down

As explained above, though, it is rarely useful to use the option of separating the right and left keys, so only the first three are generally relevant.

In each row, there is an entry for each of the modifier keys, utilising the symbol for that key, plus the letters L or R for left and right modifiers, respectively. The symbol may be blue, meaning that the modifier has to be down, red and strikethrough, meaning that the modifier must be up, or missing, meaning that it does not matter whether it is up or down.



One important thing to note first is the first column, labelled Index. In many keyboard layouts, you will see the same number repeated in that column. When there is just one row with that number, then that is the only combination of modifier keys that matches that set. If there are two or more rows with the same set number, then any of the modifier key combinations with that set number is considered to be part of the same set.

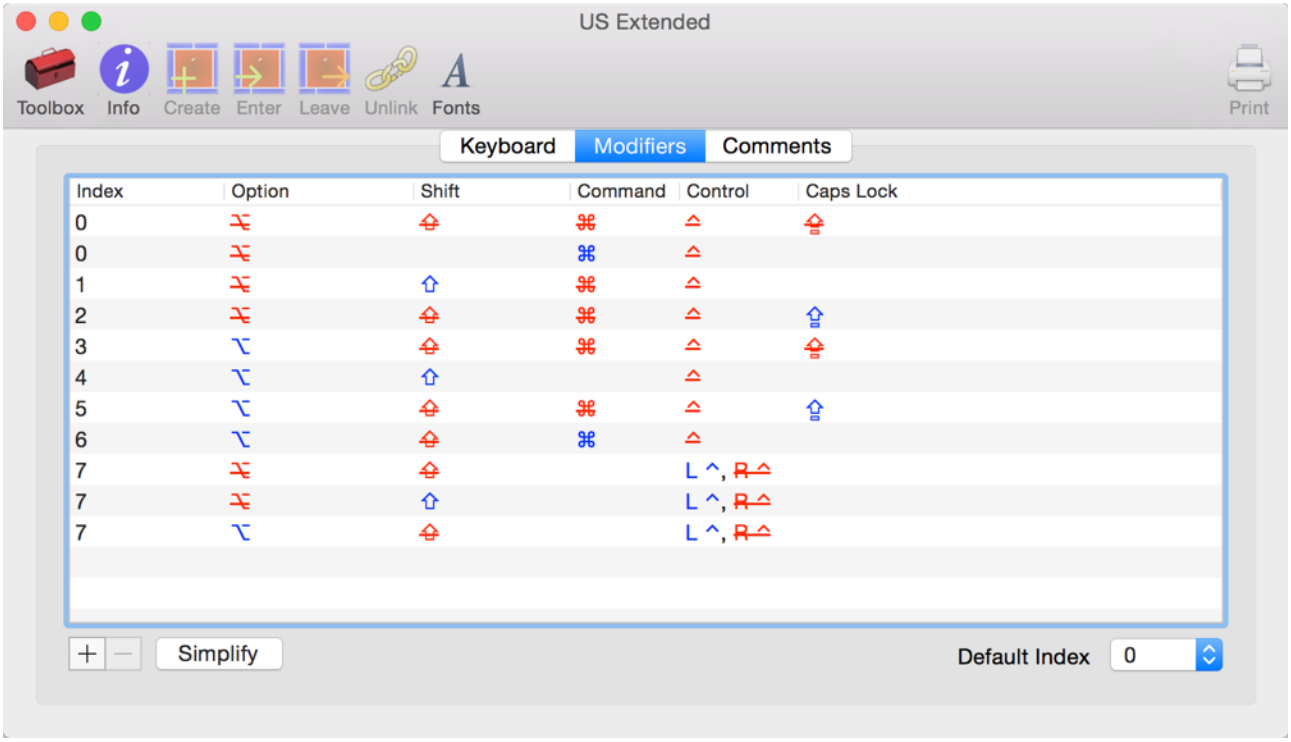
The key is to read each row independently. What the operating system is looking for is a "match". What that means is that the current state of the modifier keys is checked against each row in the table, and the last one that works is the match. As an example, consider the following set of modifier key combinations:

Index	Option	Shift	Command	Control	Caps Lock
0	Both up	Both up	Up	Both up	Up
1	Both up	Either down	Up	Both up	Down or up
2	Either down	Both up	Up	Both up	Up
3	Both up	Either down or up	Up	Both up	Down
4	Either down	Either down	Up	Both up	Down or up

If the current state of the modifier keys is that the left shift key is down, the right option key is down, and the other modifier keys are up, then the match is going to be with the row with index 4.

But what if the current state was that the command key was down? None of the rows match that, since they all specify that the command key is up. This is where the default index comes into it. You'll see the default index in the popup menu at the bottom of the window. When no row matches, then the default index is what the operating system uses. If the default index were 0, then any combination where the command key is down would be counted as a match for row 0.

One thing that is not so obvious is that there might be more than one row that matches a given set of modifiers. The important thing to note is that in this case it is the *last* matching row which is the one that is considered to be the match.



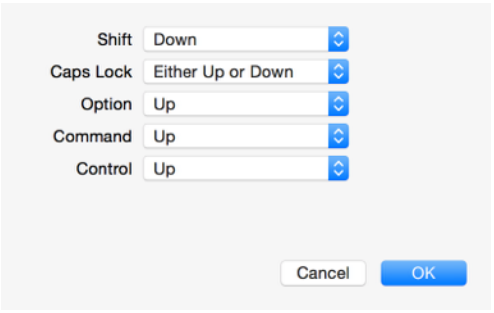
Let's look at a more complex set of modifier key combinations, that defined by the US Extended keyboard layout which is included with Ukelele:

There are quite a few things to notice about this set. One is that you will see that the last three rows include combinations which distinguish between the left and right modifier keys for control. As was said earlier, it is not usually helpful to define such combinations, as most keyboards treat both keys as left keys. However, these actually work, because they use the left key as the relevant one.

A second important thing to see in this set is that there are two rows with 0 in the first (index) column, and three lines with 7 in the first column. This means that there are two distinct combinations which match index 0, and three that match index 7. Looking at the two rows with index 0, we can see that they match either all modifier keys up (the first row) or the command key down with the shift or caps lock keys either down or up (the second row).

Finally, you will see that the Simplify button is active. If you click it, you will get simplified modifiers, which means that the three combinations with the left control key will be simplified to work with either control key.

6.8.3.2. Editing a modifier key combination

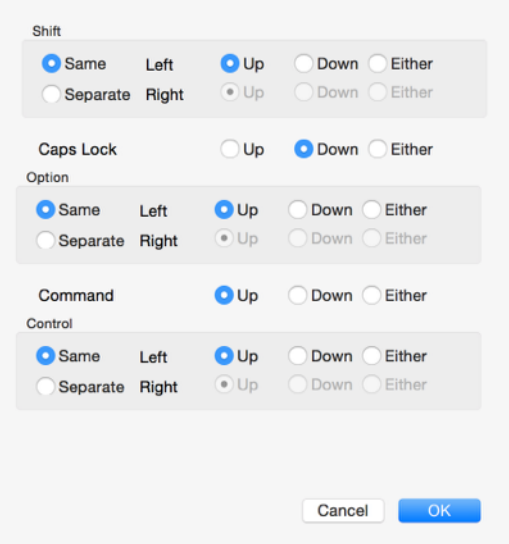


To edit an existing modifier key combination, you double-click the row in the table in the modifiers drawer. You will then see a dialog box which allows you to edit the combination. There are actually two different versions of this dialog, depending on whether the modifiers are simplified (no reference to left or right modifiers) or not (at least one reference to a left or right modifier).

The dialog for simplified modifiers is quite straightforward, offering the choice of Up, Down or Either Up or Down for each of the five modifiers.

In the dialog for non-simplified modifiers, there is a separate section for each of the five kinds of modifier key. For those that have left and right keys, that is, shift, option and control, there is a radio button for whether they are considered the same or separate. Once again, you

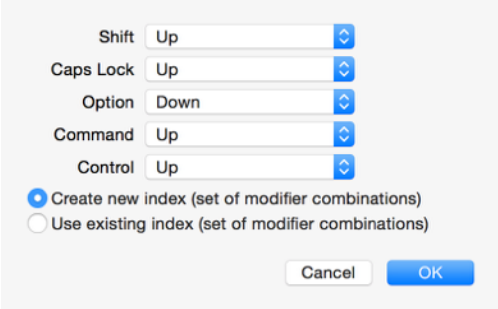
would normally have "Same" selected, as most keyboards consider both keys to be the same, so that both shift keys are considered to be left shift keys. If you really want to have separate left and right modifier keys, you can do it.



For each modifier key, you can specify one of Down, Up or Either. These are hopefully obvious, but Down means that the key has been pressed, Up that it has not been pressed, and Either that it doesn't matter whether it has been pressed or not for this combination. One exception to the definition is the caps lock key, which acts like a toggle: one press sets it as down, and a second press sets it as up, so that it is not whether the key is physically down, but the state that it is in — down is often shown by a light being on.

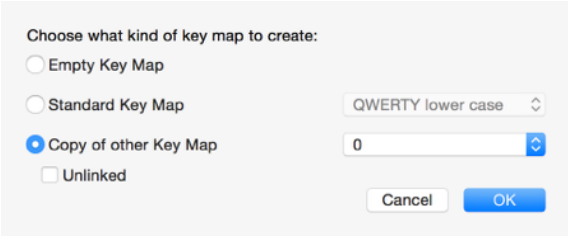
6.8.3.3. Adding a new modifier key combination

When you want to add a new modifier key combination, you have to make a decision before you start: Is it going to be a completely new modifier key combination, or is it going to be an alternate for an existing combination? In other words, is it going to appear with a new index number, or is going to be a row with the same index as an existing row? If it is going to be a row with the same index as an existing row, then you should select the row with the desired index. Otherwise, the selection does not matter. Either way, you add a modifier combination by clicking on the "+" button on the lower left of the window. This will bring up the same dialog as editing a modifier combination, but with a pair of radio buttons at the bottom.



If you want to create the new modifier key combination as an alternate for an existing row, first select that row in the table, and then choose the "Use existing index" radio button.

If you want a completely new modifier key combination, choose the "Create new index" radio button. (If you had no selection, this will be selected and the radio button disabled.)



Once you have chosen your modifiers, click OK. If you are adding an alternate, you are done. Otherwise, you will be presented with another dialog that asks what kind of key map the new modifier combination will trigger. The options are an empty key map (only special key output defined), a standard key map (see the sidebar "What are the standard key maps?"), or a copy of an existing key map. In the last case, you can pick which one, and whether it will be unlinked.

Key maps can be linked or unlinked, as can individual keys. For example, it is common for the alphabetic keys to be linked so that shift and caps lock produce the same output. If they are linked, then changing one changes the other as well. If they are unlinked, they can be changed independently. If you make the new key map linked, then you can unlink those keys that need to be, while leaving others linked. If you make it unlinked, then every key can be changed independent of how the original key map had them.

What are the standard key maps?

There are six standard key maps. Each of them defines the basic output for all the special keys, everything except the alphabetic, numeric and punctuation keys, which vary according to the standard. Many of the standard key maps are named after the first six characters on the top alphabetic row.

The empty key map does not have anything for the ordinary keys.

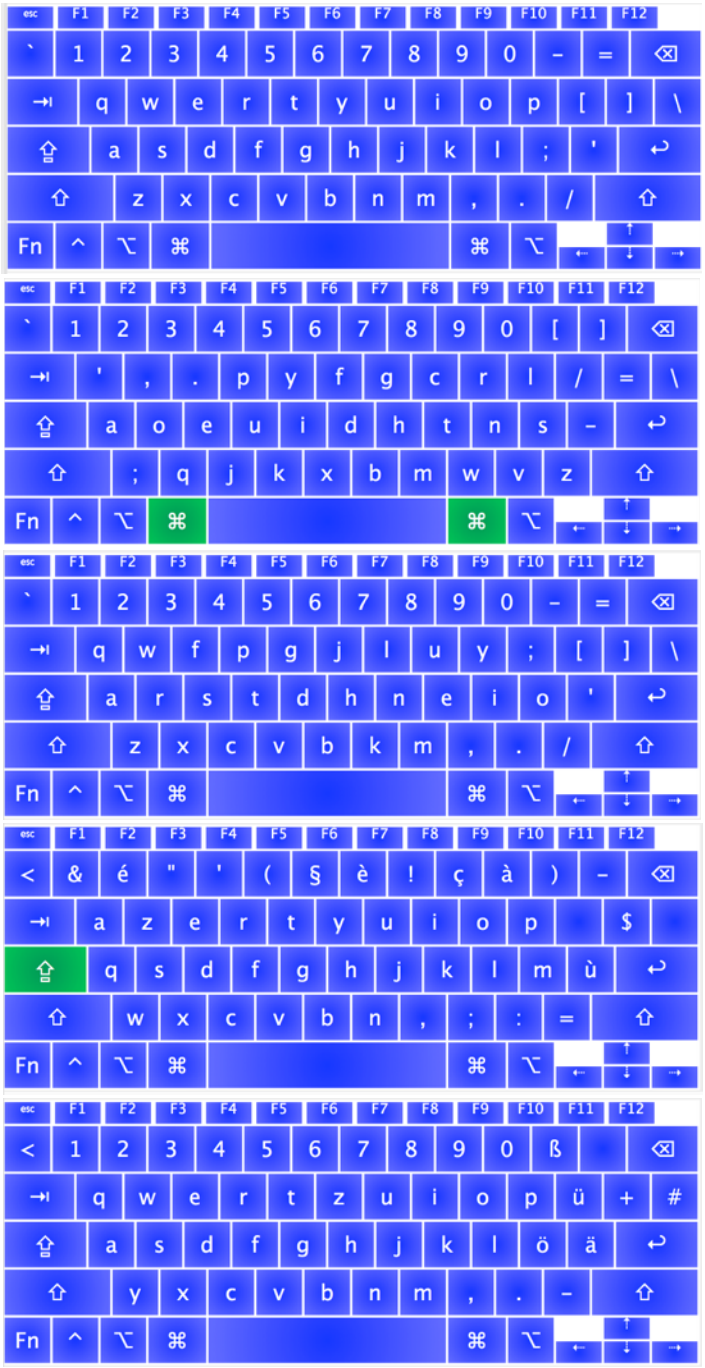
The QWERTY key map is the standard English key map.

The Dvorak key map is an alternate English key map, designed to be easier to type on.

The Colemak key map is another, more recently designed alternate key map, designed to be ergonomic.

The AZERTY key map is the standard French key map.

The QWERTZ key map is the standard German key map.



6.8.3.4. Deleting a modifier key combination

To delete a modifier key combination, first select the appropriate row in the window, and click the "-" button at the bottom of the window.

6.9. Installing a keyboard layout

When you have created a keyboard layout, you need to install it before it can be used. There are two ways to install a keyboard layout, from within Ukelele or manually. If you are developing a keyboard layout for your own computer, you can do it within Ukelele. If you are developing it for distribution to others, it will involve manual installation on other people's computers.

6.9.1. Choosing the folder for installation

There are a number of different folders where you can put your keyboard layout. Where you put it affects which users can use it. All of the folders are called Keyboard Layouts, and reside

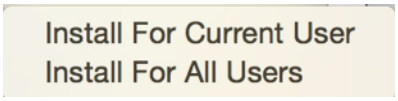
inside a folder called Library (or its localised equivalent, depending on what language your computer is set to use). It is the location of the Library folder that is important.

If you use the Library folder inside your home directory, then only you will be able to use the keyboard layout. Other users on your computer will not be able to use it. If yours is a single-user computer, then this is the simplest place to put it. This folder would be called "~/Library/Keyboard Layouts". Note that if you are using Mac OS X 10.7 (Lion) or later, the Library folder is hidden by default. Press option and select the Go menu in the Finder to access the Library folder.

If you use the Library folder that you see in the Finder when you double-click your start-up hard disk, then all the users on your computer will be able to use your keyboard layout. You must supply an administrator password to put the keyboard layout into this folder. This folder would be called "/Library/Keyboard Layouts".

If your computer is a server attached to a network, then you can also use the Library folder in the Network folder at the top level of your hard disk, that is, double-click your start-up hard disk, find the Network folder there, and use the Library folder inside that. This will allow any user connected to your computer to use your keyboard layout. The folder will be called "/Network/Library/Keyboard Layouts".

Installation from within Ukelele can only be to the first two folders. To install in the network folder, you will have to do it manually.



To install the keyboard layout on your own computer, you use the Install submenu of the File menu. It has two options: Install For Current User, which does not require a password, and Install For All Users, which requires an administrator password. The first time that you choose Install For All Users, you will be prompted for a password to install a helper application, and then again to install the keyboard layout.

6.9.2. Keyboard layout collections

Strictly speaking, only the keyboard layout file needs to be installed. It is the XML file that is created by Ukelele, and has an extension of .keylayout. It contains all that is necessary for the operating system to load the keyboard layout.

The difficulty that you may find with just installing the keyboard layout file is that you will get a generic keyboard icon in the input menu. If you want to have a more distinctive icon, such as a flag, you have two options. One is to create an icon file with the same file name as the keyboard layout file, but with the extension .icns rather than .keylayout. The other is to create a collection, in a special folder layout called a bundle.

Another issue is that support for "press and hold", introduced with OS X 10.7 (Lion), requires a keyboard layout collection rather than just the keyboard layout file. If you are installing a keyboard layout on OS X 10.7 or later, you probably should use a collection. For this reason, Ukelele 3.0 makes bundles the default format for saving a keyboard layout.

A bundle is a folder with a particular structure and contents which is presented to the user as a single file by the Finder. There are many types of bundles on a given system, including applications, plug-ins, input managers, libraries, and keyboard layouts. A keyboard layout bundle contains at least one XML keyboard layout file (or old-style resource-based keyboard layout file), and optionally icon files to go with them. You can also have localised names for any of the keyboard layouts in the bundle. We call such a bundle a keyboard layout collection.

The advantages of a collection are that you can put the icon together with the keyboard layout, you can specify the language for the keyboard layout, and you can have more than one

keyboard layout in the one location. This makes installation easier, as well as distribution to other users.

6.9.3. Manual installation

To install a keyboard layout manually, or to install it to another person's computer, is a little complicated. You have to choose whether it will be available for just the one user or for all users of the computer, or even for the network if this is a server computer.

For just the one user, the keyboard layout has to be installed in the "Keyboard Layouts" folder inside the Library folder. The Library folder is hidden by default, so you need to hold down the option key and choose Library (or its equivalent in the primary language of your Mac) from the Finder's Go menu. It's likely that the Keyboard Layouts folder doesn't exist, and has to be created. Make sure the name is correct: two words, and Layouts, not Layout.

For all users of the computer, it needs to be put in the Keyboard Layouts folder inside the Library folder at the top level of the startup disk. You will need an administrator password to make changes here. Again, the Keyboard Layouts folder may not exist, in which case you need to create it.

In current versions of Mac OS X, you do not need to log out and log in again. Rather, your keyboard layouts should be immediately available in System Preferences.

One word of caution: *Never* try to edit a keyboard layout when it has been installed. *Always* work on a copy, and then install the copy to replace the current one. Editing an active keyboard layout can cause your computer to crash, in some cases to the point that you need to work from the recovery partition to get things back to normal.

7. Miscellaneous tasks

There are various kinds of things that need to be done from time to time. Most of these are maintenance tasks, but also there are some others.

7.1. Comments

It is possible to put comments into any XML file, and keyboard layouts are no exception. Why would you want to put a comment there? One reason is to put a copyright notice there. Another would be to acknowledge sources or to document decisions, changes or limitations. They are just comments, so it's up to you!

There are actually one or two comments inserted automatically by Ukelele. If you created the keyboard layout with Ukelele, a comment saying when you created it, and with which version of Ukelele, will be inserted. Whenever you save the file, a comment saying when it was saved, and with which version of Ukelele, will be inserted, though you can turn this off in Ukelele's preferences. You can't change these comments, but you can see them. If the keyboard layout was generated with one of Apple's tools, there will be a comment saying so.

Comments are edited on the comment tab of the keyboard window. This shows the comments in the keyboard layout's XML file, with the first comment, if any, visible. You can move around the comments, add a comment, and edit or delete a comment. The only limitation is that you can't delete or edit the automatic comments.

You can also add a comment to a particular key, which might be useful for documenting what the original output was, and how or why it was changed. When the keyboard tab is active, you can select a key, then you choose "Add Comment..." from the Keyboard menu. The comment tab is activated, and the editor opens with a blank comment, which will show the key element in the bottom pane.

7.2. Changing dead key state and action names

Dead key states always have a name. Much of the time, you don't need to know the name. When you create one, Ukelele creates an automatic name unless you supply a name in the dialog. If you are dealing with dead keys a lot, particularly multi-level dead keys, you may want to change the name to something memorable.

One other reason that you may need to change a dead key state name is if some of the dead key states have names that are numbers, that is, only consist of digits. Due to a bug in the keyboard compiler that the operating system uses to load a keyboard layout, such names can cause the keyboard layout to be unusable. This is discussed more in chapter 10 on Troubleshooting.

There is also something called an "action". This is a set of possibilities for a particular key. They take the form of a set of different behaviours based on the current dead key state. For each dead key state, it can specify either what the output is, or what the next dead key state would be.

Actions also have names, but you will most likely never see them. You can change them, if you wish.

Changing dead key state names and action names are very similar tasks. You choose either "Change State Name..." or "Change Action Name..." from the Keyboard menu. You get a dialog in either case, showing a pop-up list of all the dead key state names or action names. Choose the one you want, and click OK. You then get to enter the new name. If you choose a name that already exists, you will be told so, and offered the chance to try a new name.

7.3. Removing unused dead key states and actions

After you've done some editing, particularly if you've started from an existing keyboard layout, you may end up with some dead key states that are no longer used. This doesn't really hurt, but just adds a bit to the file size of your keyboard layout. You can remove them by choosing "Remove Unused States" from the Keyboard menu.

A word of caution is warranted here. There may be a good reason for a dead key state to be unused at any given time, when you will still need it. An example is when you want to move a dead key — you can do that by changing the existing dead key to be an ordinary key, then creating a new dead key and specifying the name of the old dead key state. In between those two actions, the dead key state is unused, and would be deleted if you chose this command.

You can also remove unused actions, by choosing "Remove Unused Actions" from the Keyboard menu. This is less likely to cause you problems than removing unused dead key states.

7.4. Adding special key output

Special keys are those keys that are not modifiers, yet normally do something other than produce regular output. These include some that have visible output, like tab and return, as well as others that do other things, such as escape, delete, arrow keys, function keys, and so on.

Special keys produce their effect by producing particular non-printing characters, usually with code points less than 32 (apart from the delete key, which produces code 127). In general, you don't want to change these. Because of various design decisions made by different programmers, it often doesn't work to change them around, so that you can't reliably change the arrow keys into something else, for instance, or make something else into an arrow key.

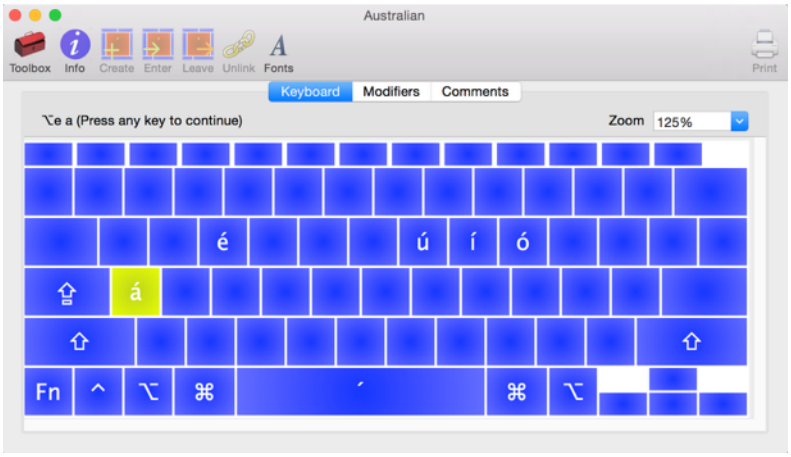
Since the special keys generally have fixed, conventional output, Ukelele allows you to set them all. When you create a new keyboard layout from scratch in Ukelele, the special keys already have their output correct. If you create one as a copy of another keyboard layout, and the original is missing output for one or more special keys (very common, given the new keyboards with F16 to F19 keys), Ukelele asks if you want to add the missing output. If you say no, you can add it yourself later.

To add special key output when you want it, just choose "Add Special Key Output" from the Keyboard menu. It will add standard output only to those special keys which don't have output already specified, so it won't undo any of your work.

7.5. Searching for an output string

Sometimes when you are working on a keyboard layout, you may not be sure whether you have already assigned a key sequence for a given output. For example, if you are updating a keyboard layout that you or someone else has created, you may wonder if a certain character has already been enabled, such as a spacing modifier or a combining diacritic. In this sort of case, the Find command is what you need.

The Find Key Stroke... command (in the Edit menu), or its keyboard equivalent ⇧⌘F, allows you to enter a string, and search for a key sequence that produces that string. Note that it has to be an exact match. There is no facility for searching for a substring. You can enter the string using the usual notation, either entering characters directly (typed, pasted, or entered via the character palette), or using the XML notation like &#01fa; for Ǻ (Latin A with ring and acute).



When you click OK (or press enter or return), Ukelele searches for a key sequence that produces that string. If there is more than one key sequence, Ukelele uses some heuristics to choose which one (shorter strings are preferred, fewer modifier keys are preferred). If there is no key sequence that produces the string, Ukelele will tell you. Be aware that this may be because that string is produced by a dead key state, but there is no key sequence to get you into that dead key state at the current time.

The mechanism for showing you the key sequence is twofold. First, the keyboard window will temporarily go to the appropriate dead key state, the relevant modifiers will appear as pressed, and the key will be highlighted (using the “selected” colour). Second, the key sequence will be shown in a text field above the keyboard layout, along with the instruction “Press any key to continue”. Both last until you press any key, when things return to normal.

Two things may cause you not to see a change in the keyboard window. One is that the string is not output by any key sequence. In this case, Ukelele will tell you that no key stroke produces that output. The other is if the string is the terminator for a dead key state, but no other key sequence produces the string. In this case, the status bar will tell you the name of the dead key state for which it is the terminator.

8. Reference

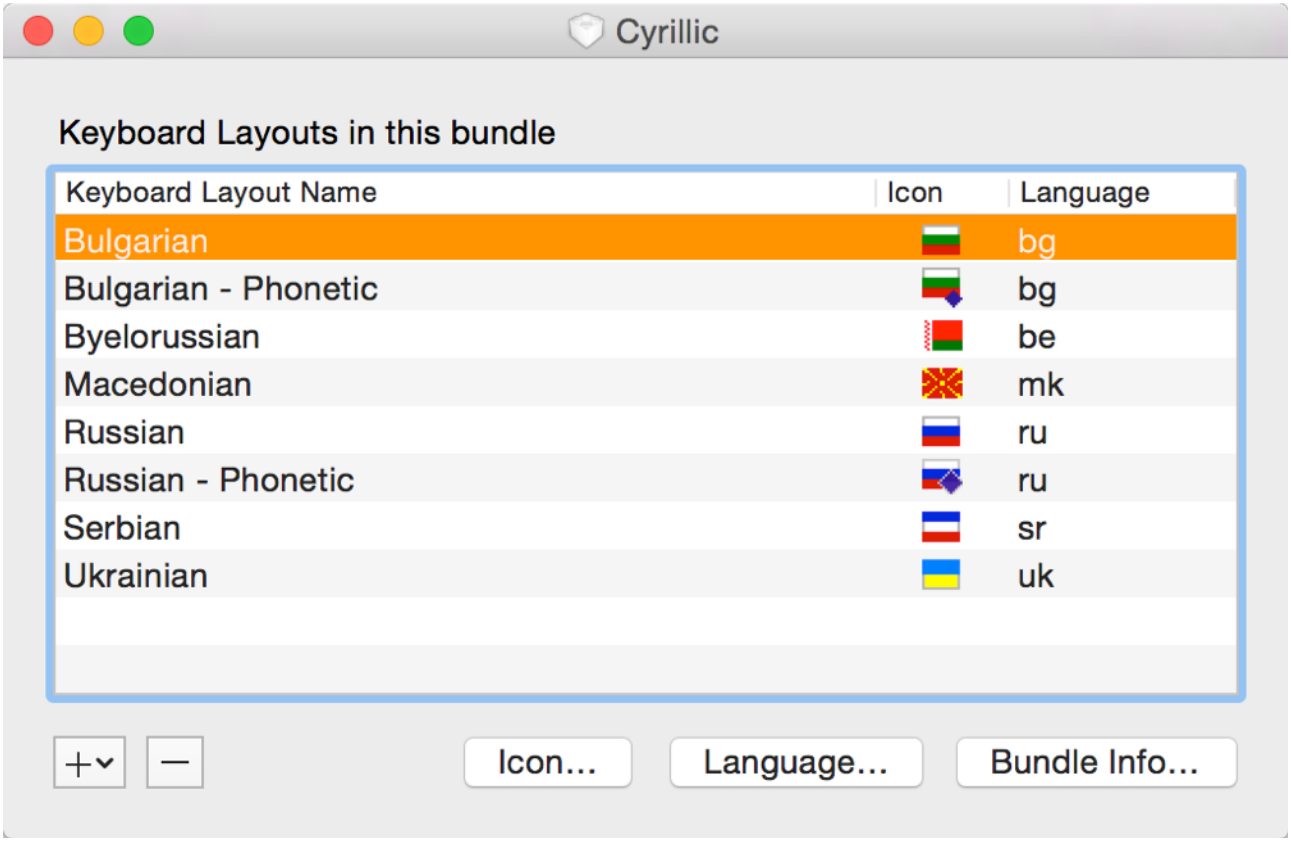
In this chapter, we will run through all the various parts of the program. We will start with all the windows, then the menus. The earlier chapters were more task-oriented, whereas this chapter is to explain all the different parts of the user interface.

8.1. Windows

Ukelele has two main window types, one for keyboard layout collections and the other for keyboard layouts, whether inside collections or stand-alone. There are also two floating windows: the info inspector and the toolbox.

8.1.1. Keyboard layout collection window

The keyboard layout collection window has a table listing all the keyboard layouts contained in the collection, along with information about them. Then there are some buttons below this table with various functions. The collection is technically a bundle, a folder packaged to appear like a single file in the Finder, and it can contain one or more keyboard layouts.



The table has three columns: the keyboard layout's name, the icon associated with it (if any), and the language code associated with it (if any). The table can be sorted by keyboard layout

name or by language, ascending or descending, by clicking the column headers, as is standard for such tables.

The keyboard layout's name is the one that will appear in System Preference's list of keyboard layouts, and in the input menu. This is set from within the keyboard layout, and is not tied to any file name. You can change the name via the info inspector, or by opening the keyboard layout window and setting the name via the Keyboard > Set Keyboard Name and ID... command.

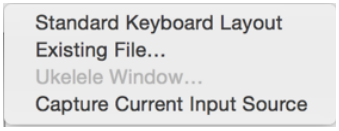
The icon is optional, but is useful to see that your keyboard layout is active. If there is an icon for an active keyboard layout, it is displayed in the input menu, and otherwise a generic keyboard icon is used.

The language is also optional. This sets the "intended language" for the keyboard layout, which is used by the operating system to know what to display for "press and hold". Given that this is built into the system by Apple, with no public method of customising it, this is limited to those languages supported by Apple. There is a list of supported languages in the appendix.

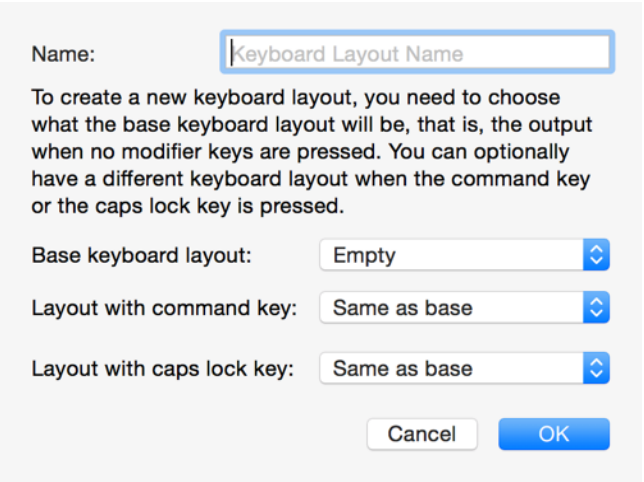
You can drag a keyboard layout from one collection to another. The icon and language attributes are retained when you do so. You can also drag a keyboard layout from the Finder to the window to add it to the collection. Dragging a keyboard layout to the Finder copies the keyboard layout file to the Finder. Dragging an icon file onto a keyboard layout sets the icon for the keyboard layout.

Below the table are several buttons. The first two are for adding and removing keyboard layouts from the collection. The other three are for setting the icon or language of a particular keyboard layout, and for setting some information about the bundle itself.

The + button has its own menu with four options, three of which are always active, and one which is only active in certain circumstances.



Standard Keyboard Layout allows you to create a basic keyboard based on one or more of several standard keyboard layouts. The standard keyboard layouts are as discussed in the sidebar on page 37 above: empty, QWERTY, QWERTZ, AZERTY, Dvorak and Colemak.



You choose a base keyboard layout, which is the keyboard layout that is active with no modifiers pressed. You have the option of having one or two additional keyboard layouts, either with the command key or the caps lock key active. This allows you to have, for example, a Dvorak layout which turns into a QWERTY layout when the command key is down, helpful for using command key combinations.

The Existing File... item brings up a standard open file dialog, which allows you to add a keyboard layout from an existing file. Note that you cannot add another collection, but only a non-bundled keyboard layout.

The Ukelele Window... item is only enabled when there is a keyboard layout window open for a keyboard layout that is not in a collection. When you select it, you will be asked to choose which of the possible windows you wish to copy into the collection.

Capture Current Input Source allows you to turn the current keyboard input source (that is, input methods other than keyboards are excluded) into an editable keyboard layout. This will

attempt to copy the icon and language attributes, if present in the current keyboard input source, and makes the new keyboard layout have the same name as you would see in the input menu.

The Icon... button opens a standard open file dialog that allows you to add an icon to the keyboard layout. The file you choose has to be in Apple's icns format, with a file extension of .icns. The icon should have at least a 16 × 16 size, and preferably a 32 × 32 size as well for use with Retina screens. You can also add an icon by dragging an icon file to the keyboard layout in the table.

The Language... button allows you to specify the "intended language" for the keyboard layout. This is used by the "Press and Hold" facility in Mac OS X 10.7 and later. Only a limited number of languages are actually supported by Apple, and these are listed in the appendix. In most cases, you would need to specify only the language, but script may be required in a couple of cases. See the fuller discussion in section 6.4.1.

The Bundle Information... button allows you to set the name of the bundle and some version numbers. This information is not visible to the user, but is for you to use in whichever way you wish to.

8.1.2. Keyboard layout window

This is the main window for editing keyboard layouts with Ukelele. It comes with a tabbed interface, allowing you to deal with the keyboard, the modifiers, or the comments.

8.1.2.1. The keyboard tab

The window shows the current keyboard layout, using a particular hardware keyboard type. By default, this is the hardware keyboard type that the operating system reports as being active on the computer.

You can modify what keyboard type is shown in the window by using the "Set Keyboard Type..." in the Keyboard menu. Be aware that this *only* affects how the window appears, and has no effect on the actual keyboard layout.

Within the keyboard layout window, there is a representation of every key on the keyboard, each of them showing what the output is for that key. This is dynamic, changing as you press or

release modifier keys. If you hold a key down, that will also be shown in the window. Pressing or clicking a key toggles its state as selected or not.

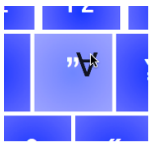
Some keys do not show their output directly, but show it by a different representation. For special keys (escape, delete, tab, return, enter, arrow keys, function keys, etc), a symbol showing their function is shown, as long as the standard output is set for the key. The modifier keys show a symbol for their function as well. A key that produces output that falls into one of the control ranges of Unicode (C0 controls are U+0000 to U+001E, C1 controls are U+0080 to U+009F), or is an invisible character (non-standard spaces, soft hyphen, etc) will be shown as its hex representation, such as U+13 for the "device control three" control.

The Unicode standard provides a range of "combining diacritics" which will attach themselves to another character. In Ukelele, these are shown with your choice of character, a dotted square or circle, a bar or rectangle, or a space. The default is the dotted square, but the choice can be made in the preferences. Note however, that not all fonts support combining diacritics correctly, and applications need to know how to handle them, too.

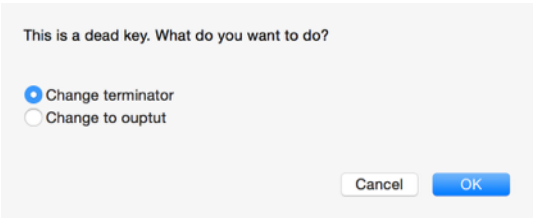
When "sticky modifiers" are turned on, then the modifier keys behave like caps lock, with one press making them stay down, and a second press making them come back up. Clicking on a modifier key in the keyboard layout window works the same as pressing the equivalent modifier key.

A key shown with the dead key colour scheme (red when not selected, yellow/orange when selected) is a dead key. See elsewhere for a full discussion of dead keys, but the basic definition is that a dead key produces no output of its own, but affects the following key press.

Within the keyboard layout window, you can change the output of a key by dragging and dropping a Unicode string (one or more characters, up to 20 UTF-16 code points) from a source such as the Character Viewer or another Unicode-aware application onto the appropriate key. The key will show the drag highlight as you drag onto it, and the output will change when you release the mouse to complete the drop. The modifiers that are pressed when the mouse is released are those that are relevant. In other words, if the option key is down when you drop the new output onto a key, it changes the output for that key with the option key down.



If you drag text onto a dead key, you are offered two options: you can make the dragged text replace the terminator for that state, or make the dragged text become the new output of the key, thus changing it into an ordinary key rather than a dead key.

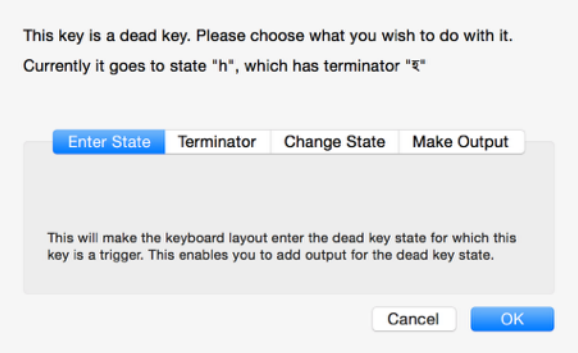


Double-clicking a key brings up a dialog that allows you to enter new output for the key. This can be a popover (as shown) or a sheet dialog.

You can use this mechanism to make a key produce a character that can't be dragged and dropped, such as a control (C0 or C1 controls), by specifying the output as a numeric string. The format for the number is very precise. It must be of the form &#<number>; (where <number> is a decimal number) or &#x<number>; (where <number> is a hexadecimal number). Note that it must be a lowercase x for the hexadecimal number, but the number can consist of 0–9, a–f and A–F. Examples are ½ or ½ or ½ or ½ each representing U+00BD, ½ (vulgar fraction one half), or Γ or Γ for U+0393, Γ (Greek capital letter gamma).

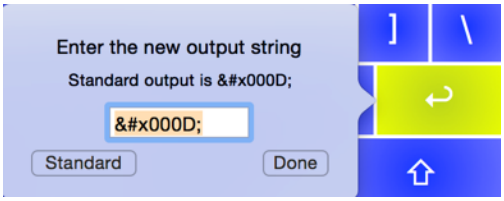
If you double-click a dead key, you are offered four choices: entering the dead key state, changing the terminator of the dead key state, changing the dead key to trigger a different dead

key state, or changing the dead key into an ordinary key and assigning new output. You choose one of the tabs and then fill in whatever is necessary to make the change you want.



In the preferences, you can select a check box that will allow you to edit the output of a key with a single click rather than a double-click. All these instructions are the same, with a single click instead of a double-click.

The keys that are classified as “special keys” are those that do not normally produce regular output. These are escape, tab, delete, return, enter, arrow keys, page up, page down, home, end, clear, help, the function keys (F1 to F19), forward delete, two Japanese conversion keys, and a few keys which have key codes that don’t appear to be used on any Apple keyboard. Special keys do not accept drag and drop for changing their output. You must double-click these keys to change their output. When you do, the popover will list the standard output of the key. An easy way to enter the standard output is to click the Standard button. Be aware that changing the output of one of these special keys may not work as you hope, due to some limitations in the way that Apple implements keyboard input handling.



Most operations that work on a particular key expect you to select the key first, then choose the command from the menu or click the toolbar button. If no key is selected, then you will be asked to select the key after you have chosen the command. In the case of swapping keys, you need to select two keys, so you should select one before choosing the command, and the second after you are prompted. If you don’t have a selected key, you will be prompted to select the two keys.

In the top right corner of the keyboard tab is a pop-up menu labelled Zoom, which allows you to set the scale of the keyboard layout. You can choose from one of the preset scales, set it to fit the keyboard layout on the screen, or set it to any particular percentage between 100% and 500%.

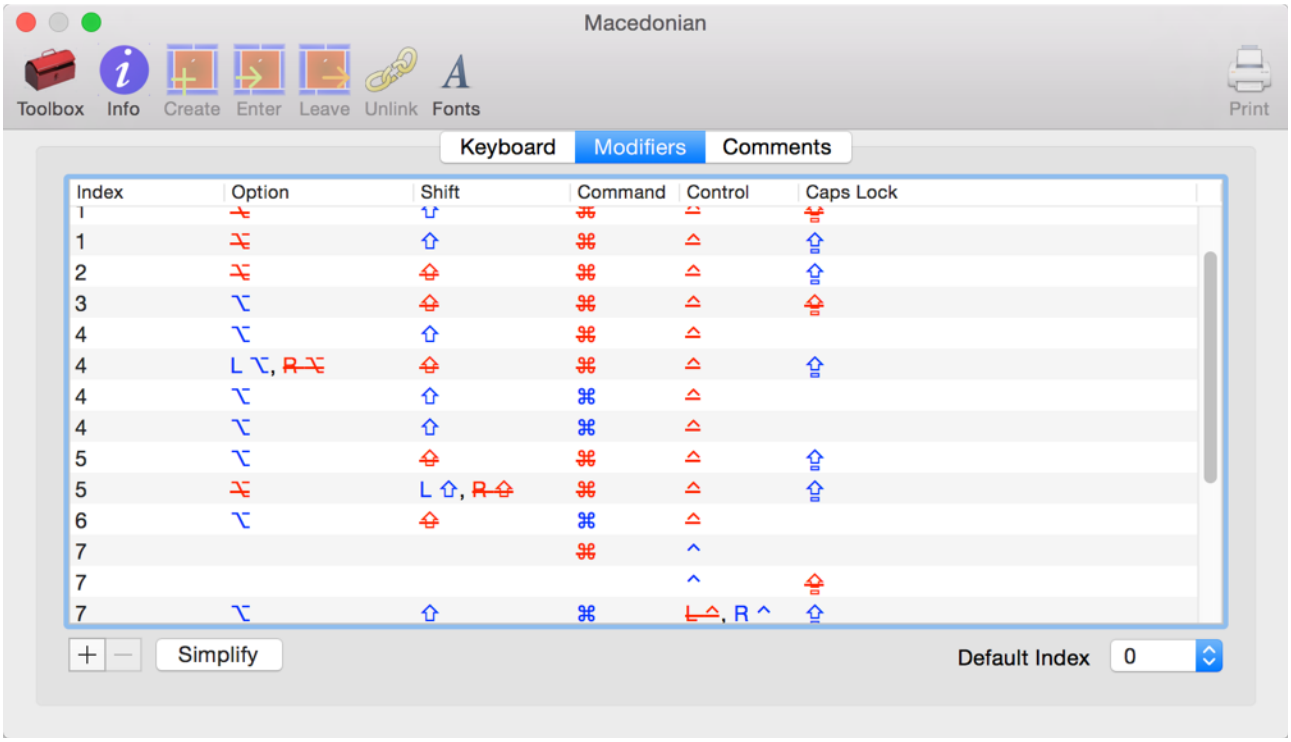
You can also change the zoom by using the pinch or zoom gestures on a compatible device (trackpad, multi-touch mouse, etc). The range is again constrained to within the range 100% to 500%.

Above the keyboard on the left is where any prompts for you to do something will appear. Most of the time, this will be empty.

8.1.2.2. Modifiers tab

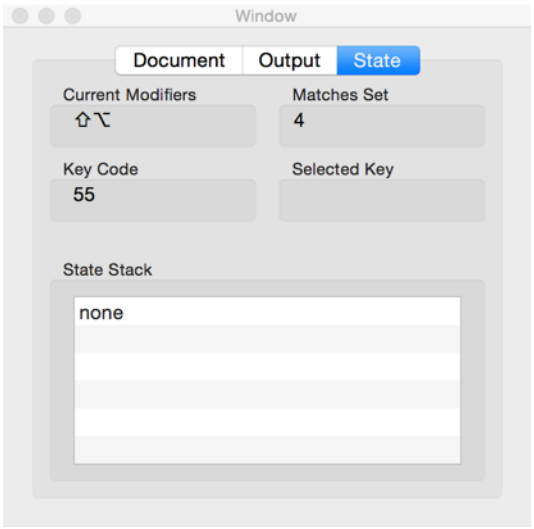
Modifier combinations are examined and edited from the modifier combinations tab. This presents a table of all the modifier combinations, organised by the *index*, which groups together sets of modifier combinations to reference the same key map. In the Macedonian keyboard layout in the figure, you can see four rows with index 4, and two rows with index 5. These refer to different combinations that mean the same when calculating what output to produce.

Modifier keys are shift, option, command, control and caps lock. In theory, shift, option and control have separate left and right keys. In practice, however, Apple currently produces no keyboards that distinguish between the left and right modifier keys, with both shift keys indicating to the system that the left shift key is down, for example.



In principle, up to 32 combinations of modifier keys can be specified, but it is rare to need that many. Most keyboard layouts define from five to ten different modifier key combinations. See the discussion above in section 6.8.3.1.

Each row in the table represents a combination of modifier keys. Each of the five modifiers can be specified as up, down, or either down or up, the last meaning that it doesn't matter. The three modifiers that have left and right keys can be specified separately, giving nine possibilities: both up, both down, both either up or down, left up and right down, left down and right up, left up and right either down or up, left down and right either up or down, left either up or down and right up, and left either up or down and right down. The way that these are displayed is using symbols, to make the table less cluttered. For each modifier, blue text signifies that it must be pressed (down). Red text with strikethrough means that it must not be pressed (it must be up). The absence of the modifier indicates that either pressed or not pressed (down or up) will match. For those modifiers with separate left and right keys, a letter is prefixed, L or R, to indicate left and right, respectively.



In the bottom right hand corner is a pop-up button giving the default index. If the combination of modifiers that are held down does not match any row in the table, then the default index tells you which set the system will assign to it. Also, note that the way that the system works is that the *last* row that matches is chosen, not the first.

To help you work with modifier combinations, the inspector window's State tab shows what modifiers are currently recognised by the system, and which set they match. With the Macedonian keyboard layout, shift and option (⇧⌥) matches set index 4.

To edit a modifier combination, double-click the corresponding row. This brings up the modifier combinations dialog. There are two versions of this dialog, depending on whether the current set of modifier combinations is considered to be "simplified", meaning that it does not involve a left or right modifier key in any combination. Select the combination that you want and click OK, and the row will show the new value.

Because there is no real use in having separate modifier combinations involving the left and right modifier keys, since there is currently no way to generate a right modifier key, it is sometimes helpful to change an existing set to the equivalent simplified set. This is done with the Simplify button below the table, which is only enabled when the current modifiers set is not already simplified.

You can delete a modifier combination by selecting it and clicking the "-" button.

To add a modifier combination, you can either add one to an existing set, or create a new set. These options are chosen in the dialog that appears when you click the "+" button. You can only add a set to an existing set if you have already chosen a row in that set.

A new modifier map can be one of a set of standard maps: empty, QWERTY lower case or upper case, Dvorak lower case or upper case, Colemak lower or upper case, AZERTY lower case or upper case, or QWERTZ lower case or upper case. Alternatively, a new modifier map can be a copy of an existing modifier map in the current keyboard layout.

The standard maps define all the special keys and, apart from the empty map, define the alphabetic keys for the Roman alphabet in the standard English layout (QWERTY), the Dvorak layout (Dvorak), the Colemak layout (Colemak), the standard French layout (AZERTY), or the standard German layout (QWERTZ).

8.1.2.3. Comments tab

Any XML file can have comments, and Mac OS X keyboard layout files are no exception. Ukelele adds a comment when creating a new keyboard layout, giving a date stamp for its creation. Whenever a keyboard layout is saved, another comment is either added or updated, giving a date stamp for the time it was saved, and also listing the current version of Ukelele used for the editing. This can be turned off in the preferences.

In the comments editor, which is on the Comments tab of the keyboard layout window, these comments are shown, but cannot be changed or deleted. Other comments can be added, changed or deleted as you wish.

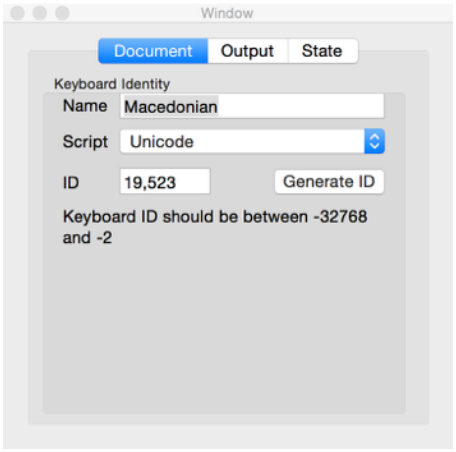
Within the comments editor, there are four navigation buttons in the top left, going to the first, previous, next and last comments. In the top right are buttons to add and delete a comment. The middle part of the window shows the current comment, and the bottom shows what element the comment is attached to.

Other comments may include when and how the keyboard layout was generated, such as from a uchr resource, or a copyright statement. You can add as many comments as you wish, for whatever purpose you wish.

You can also attach a comment to a particular key with given modifiers. You do this by choosing Keyboard > Add Comment, and then type or click the key to which you wish to add a comment. The comment editor will open a new comment, and you will see that it is attached to the key element that you chose.

8.1.3. The info inspector

The info inspector is shown by either clicking the Info button on the toolbar of the keyboard layout window, choosing View > Show Info, or pressing ⇧⌘I. It is a floating window with three tabs.



8.1.3.1. Document tab

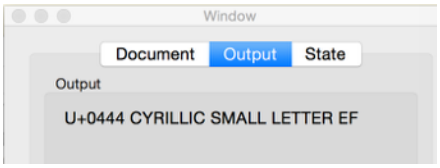
The document tab shows information about the currently selected keyboard layout. If the frontmost window is a bundle window, the selected keyboard layout's information is shown. If the frontmost window is a keyboard layout window, then information about that window is shown.

The pieces of information shown are the keyboard layout's name, which is the one that will appear in the input menu and in System Preferences, the Script associated with the keyboard layout, and its ID.

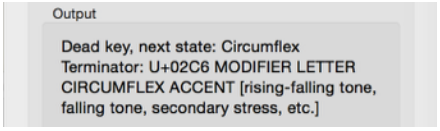
Keyboard layouts are associated with one of Apple's scripts, which are Unicode, Roman, Japanese, Simplified Chinese, Traditional Chinese, Korean, Cyrillic and Central European. In almost all cases, you should set the script to Unicode. Keyboard layouts can only generate Unicode characters within the range set by the Script, and Unicode allows all possible code points.

Each keyboard layout has an ID, which has to fall into a certain range, which is defined by the Script. Whenever you change the Script, a random ID will be generated. You can also generate a new ID by clicking the Generate ID button.

8.1.3.2. Output tab



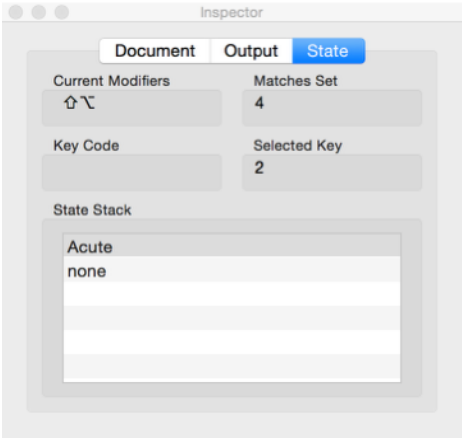
The Output tab shows the output of the key that is currently under the pointer, with whatever modifiers are current. This is given as a Unicode code point in the U+xxxx format, plus a description drawn from the official Unicode Character Database (available at http://www.unicode.org/ucd/). If you have more than one character output, some of it may be clipped if the descriptions happen to be long, though the window can be resized.



If the key under the pointer is a dead key, then the output tab lists the name of the dead key state and the terminator.

8.1.3.3. State tab

The State tab shows five pieces of information: the current modifiers, the set that those modifiers match, the key code of the key that is currently pressed, the key code of the currently selected key, and the state stack.



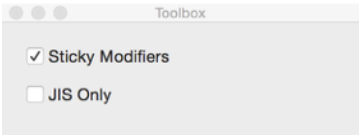
The current modifiers are those that are active, whether currently pressed or made active via the "sticky modifiers" option. The set that is matched by these modifiers is shown as well. Refer to the Modifiers tab of the keyboard layout window to see the actual modifier combinations and their set numbers (listed as Index there).

The key code of the key that is currently pressed is shown as long as the key is down. If you have multiple keys

pressed, the key code of the last key pressed is shown. Note that the modifier keys do not show their key code. If there is a selected key, its code is shown.

The list at the bottom is the "state stack". This shows the stack of dead key states. At the bottom of the stack is the special state "none", which is used to indicate that there is no dead key state active. As you enter a dead key state, its name is pushed onto the top of the stack, and popped off when you leave the dead key state. It is possible to have a stack arbitrarily deep, but you rarely go more than a couple of states deep.

8.1.4. Toolbox



The toolbox is a floating window with two controls in it. One is a checkbox for turning on sticky modifiers (which can also be done from the View menu).

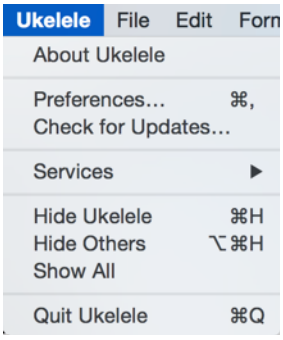
The other is a checkbox for dealing with JIS keyboard behaviour. Some types of keyboard are designated as JIS keyboards. These will be those produced for the Japanese market. They usually have some extra keys for composing Japanese characters. When the current keyboard is a JIS keyboard, the JIS palette will be shown by default.

The checkbox affects the way that changes to output are made. JIS keyboards traditionally have a small number of keys that produce different output to the corresponding ANSI or ISO keyboard. These are stored as "JIS overrides", and the JIS palette allows you to choose whether any changes are for the JIS overrides or for all keyboard types. Just choose the appropriate option: JIS Only checked means that you are editing the JIS overrides, whereas JIS Only unchecked edits the underlying output for all keyboard types.

8.2. Menus

The menus in Ukelele are mostly standard menus. The main menus that you will use within Ukelele are the Keyboard menu, which contains most of the commands for working on a keyboard layout, and the View menu, which allows you to change the way the windows appear.

8.2.1. Ukelele menu

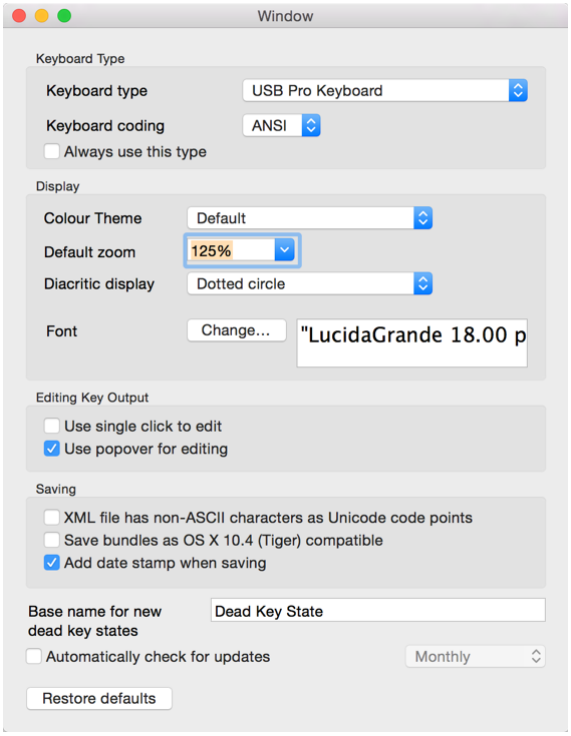


The Ukelele menu is a standard menu, containing the usual items. The only menu items that are of any significance is the Preferences... item, which has the standard shortcut of ⌘, and the Check for Updates... command.

Check for Updates... uses a different mechanism to Ukelele 2.x (the Sparkle framework, for those who are interested). It will check on the internet (if you are connected) for a newer version. If there is one, then a window will pop up showing you the release notes and allowing you to decide what to do. You can decide to skip the new version, in which case you will not see this update again, but only the following version. If you click on "Remind Me Later", the update will not be installed, and will be shown to you when you next check. If you click on "Install Update", Ukelele will download the update and install it for you. You can also opt to install updates automatically.

The preferences dialog has several sections.

The Keyboard Type section allows you to specify a default keyboard type for displaying keyboard layouts. As noted above, this only affects how it appears on screen, and has no effect on the keyboard layout file. The default is useful if you have a non-Apple keyboard attached to your



computer. You can also opt to have Ukelele use this keyboard type always, ignoring the hardware keyboard that is detected.

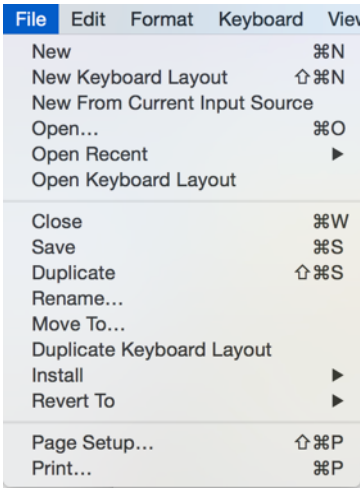
The Display section allows you to choose a few more things about the display. At present, there are only two colour themes, Default and Print, which are used for on screen display and print respectively. The zoom for a new keyboard layout window can be chosen from the popup menu, or you can type a custom figure, between 100% and 500%. The Diacritic display option allows you to choose what base character is shown when the output for a key is a combining diacritic. Finally, the font used for displaying keyboard layouts is chosen by clicking the Change... button and selecting it in the font panel. The chosen font is displayed in the panel so that you can see what it looks like.

The next section is Editing Key Output. It has two options. The first allows you to edit a key's output with a single click instead of the standard double-click. The other allows you to choose between a popover and a sheet for editing key output.

The Saving section has three options. One is to make Ukelele save its XML files with non-ASCII characters saved as Unicode code points in hex format. This makes no difference to the function of the keyboard layout, but can be useful for checking the XML file after editing it. The next option is for keyboard layout bundles, with the option of saving the bundle so that they can be used on pre-10.5 systems, that is, Mac OS X 10.4 (Tiger) or earlier. This only changes the bundle ID, which is used by the operating system to identify what kind of bundle it is. In future versions of OS X and Ukelele, this may be used for other purposes, so it is probably preferable to turn this option off unless you need to provide support for earlier versions of OS X. Finally, Ukelele adds a date stamp whenever you save a keyboard layout, but this can be disabled here.

The remaining options are the default base name for dead key states and software update. The default base name gives a string to which Ukelele adds an incrementing number to create unique new names for dead key states. For checking for updates, you can turn it on or off, and specify whether you want to check daily, weekly or monthly.

8.2.2. File menu



The File menu contains standard items as well as some items that are specific to Ukelele.

New

When you choose New (or press ⌘N), Ukelele will create a new, empty keyboard layout collection.

New Keyboard Layout

This creates a new, unbundled keyboard layout. This is an empty keyboard layout, and has only special key output, and a basic set of modifier combinations.

New From Current Input Source

This option will take the current keyboard input source and convert it to a form that Ukelele can edit. This can be a way of using the system keyboard layouts in Mac OS X 10.5 (Leopard) and later, which are not supplied as XML files.

This is also a friendly way of converting older resource-based (uchr only) keyboard layouts into a form that can be modified with Ukelele. Older KCHR resource-based keyboard layouts may not be usable on the latest versions of Mac OS X, which could make this impossible. Such keyboard layouts could be converted on older versions of Ukelele running on a PowerPC processor.

Open…

This brings up a standard open file dialog. You should only open keyboard layout files, that is, those with the extension .keylayout, or keyboard layouts collections, which are presented as bundles. It is possible to open other bundle types, but should get an error message that the bundle does not contain any keyboard layouts when you do. This is because all bundles with extension .bundle are considered alike by the operating system.

Open Recent

This submenu lists the most recent keyboard layouts that have been opened.

Close

As usual, this command closes the current window. If it is a keyboard layout window for a keyboard layout in a bundle, nothing special happens. If the window represents a new bundle that has been changed but never saved, you will be asked whether to save it first.

Save

As expected, this saves the current file, and is only enabled when the frontmost document has been changed but not saved. Note that, when the frontmost window is a keyboard layout from within a bundle, the Save command is not enabled.

Most of the time, you will not need to use this command. Saving happens automatically. You may want to save the document manually to get a "version" that you can recover if necessary.

Duplicate

This command saves the frontmost document to another file, in effect making a copy. It is only enabled when a keyboard layout document is open. It has the keyboard shortcut ⇧⌘S.

Rename…

This will change the name of the current document.

Move To…

This allows you to move the current document to a new location.

Duplicate Keyboard Layout

This command creates a copy of the selected keyboard layout within the current bundle.

Install

This submenu has two commands, Install For Current User and Install For All Users. The first installs a copy of the current keyboard layout collection or unbundled keyboard layout into the current user's Keyboard Layouts folder, which makes it available for you to use as an input source. The second command installs it in the global Library's Keyboard Layouts folder, so that all users on this computer can use the keyboard layout. It requires an administrator password to

install for all users. The first time you do such an installation, Ukelele will prompt for a password to install a helper tool, so you will see two prompts for a password.

Revert To

This is a standard dialog item.

Page Setup…

This is the standard Page Setup dialog.

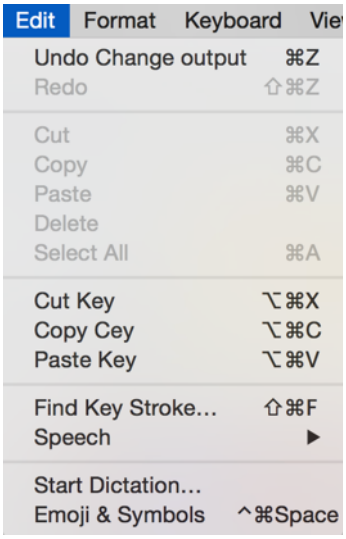
Print…

If the frontmost window is a keyboard layout bundle, this prints a list of the keyboard layouts in it. If the frontmost window is a keyboard layout window, this lets you print the current keyboard layout, using the same keyboard type as is currently shown in the keyboard window, except that it is printed with the Print colour theme.

There are two options in printing a keyboard layout. You can have the keyboard layout print the output for all dead key states, or only the current dead key state. The second option is to print the output for all modifier combinations, or for just the current modifier combination.

If you don't want to print the whole set of pages, you can create a PDF file of the print output, and just print whichever pages will be useful. This is done by selecting a PDF option from the menu that pops up from the button in the bottom left corner of the print dialog.

8.2.3. Edit menu



The Edit menu has only a few commands, most of which are standard.

Undo

Most actions in Ukelele can be undone. The menu will change to show what the last action was.

Redo

Any action that has been undone can be redone. Again, the menu will change to show what the action is that can be redone. Note that performing another action after undoing an action clears the redo stack, so you can only redo something immediately after the undo. The keyboard shortcut is ⇧⌘Z.

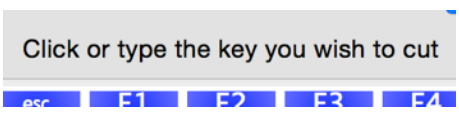
Cut, Copy, Paste, Delete, Select All

These are the standard commands, and are only enabled when a text field in a dialog is active, and (for Paste) there is text on the clipboard or (for Cut, Copy and Delete) text is selected, or (for Select All) not all text is selected.

Cut Key…

If you want to move a key to a different spot on the keyboard, you can cut and paste it. Cutting a key affects the key's output or dead key status with all modifier combinations, such as no modifiers, shift, option, option-shift, etc.

When you choose this menu item (or type the keyboard shortcut, ⌥⌘X), what happens next depends on whether you have a selected key. If so, the key is cut. Otherwise, a message appears in the message area above the keyboard layout, telling you to type or click the key.



Copy Key…

This command functions much as Cut Key..., though without clearing the key from the keyboard. It has the keyboard shortcut ⌥⌘C.

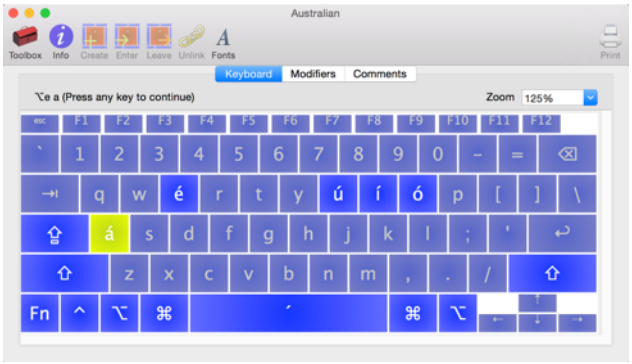
Paste Key…

This is only enabled when a key has been cut or copied with either the Cut Key... or Copy Key... commands. Like those, if you do not have a selected key, you are presented with a prompt to click or type the target key. The keyboard shortcut is ⌥⌘V.

Find Key Stroke…

The Find Key Stroke command allows you to search for keystrokes that will produce a given string. When you choose the command (or use the keyboard shortcut, ⇧⌘F), you get a dialog which asks you to specify a string. You enter the character (or characters) that you are looking for, using the same notation as used elsewhere, either literal characters or XML sequences such as ε for the Greek small letter epsilon.

When you click OK (or press enter or return), Ukelele finds a key sequence that will produce that string, and tells you what it is in the message area above the keyboard layout. If the string is not produced by any key sequence, you will be told so. Otherwise, the key sequence is shown in the status bar, and the relevant key is highlighted. Ukelele temporarily shows the dead key state and modifiers to produce the string, and returns to normal when you press any key.



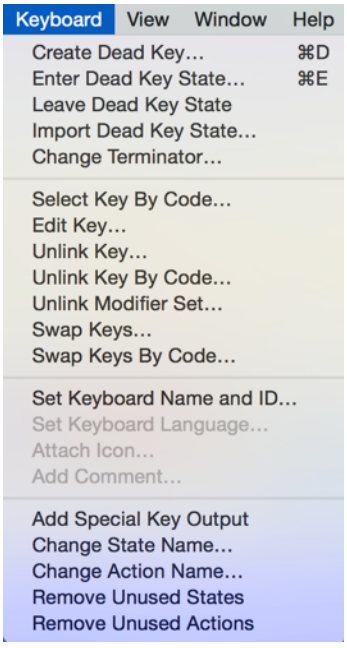
Speech, Start Dictation…, Emoji & Symbols

These are standard menu items or submenus. The last shows the Character Viewer.

8.2.4. Format menu

The Format menu is the standard OS X format menu, with submenus for Font and Text.

8.2.5. Keyboard menu



The Keyboard menu has most of the commands used for editing a keyboard layout. Some commands are only appropriate for keyboard layout windows, others for only keyboard layout collection windows, and some for both.

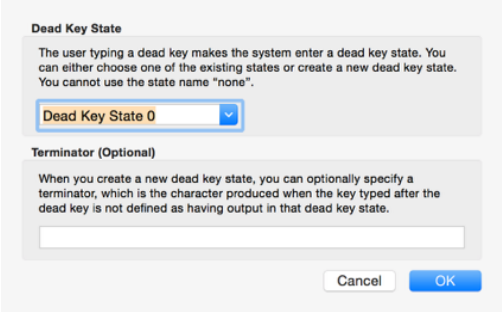
Create Dead Key…

The whole area of dead keys is complex, and is dealt with more in section 6.7. What happens when you choose Create Dead Key... depends on whether you have a selected key or not. If you have a selected key, then that key with the current modifiers will become the dead key. Otherwise, you are prompted for the dead key in the following dialog.

Next you will see a dialog appear, which will have an extra section at the top if you did not have a selected key. The common

elements are choosing a dead key state and (optionally) a terminator.

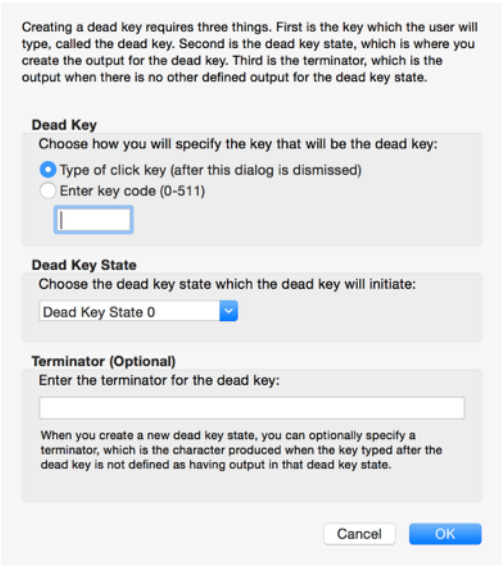
For the dead key state, the combo box allows you to choose an existing dead key state, or create a new one. The default name for a new dead key state is filled in and initially selected by Ukelele. If you choose an existing dead key state, then anything you supply for the terminator will be ignored, and the new dead key will trigger the existing dead key state.



If you give a new name for a dead key state, Ukelele creates the new dead key state, with whatever you supply for the terminator. You don't have to supply a terminator, in which case the terminator is simply the empty string. Otherwise, it is the character that will be produced if the next key stroke doesn't have any output defined for the dead key state.

Once you have created the new dead key, Ukelele will update the state stack in the Info Inspector to add the new state at the top, and the keyboard layout window will update to show the output of keys in that state.

This menu item has the keyboard shortcut ⌘D, and is also available as a button on the toolbar, called Create.



Enter Dead Key State…

If you have at least one dead key state defined in your keyboard layout, then you can choose to enter a dead key state. This means that you can edit the output of keys in that dead key state. Choosing this menu item, using the keyboard shortcut ⌘E, or clicking the Enter button on the toolbar brings up a dialog asking you to choose which dead key state to enter. If there is only one dead key state defined, then the command is disabled once you have entered that dead key state.

Leave Dead Key State

This command is only available when there is a dead key state active. What state is active can be seen in the state stack, and is the topmost state listed. If there is only the state "none" listed, then no dead key is active. It is available in the Keyboard menu and as a button on the toolbar, named Leave.

The effect of this command is to remove the top dead key state from the state stack and go to the state that is listed immediately below it. When you do this, you will see that the keyboard layout window will update to show outputs in the previous dead key state.

Import Dead Key State…

Sometimes, you want to create a keyboard layout that has some things in common with another keyboard layout, but it's not close enough that you want to base the new layout on that other layout. Or you might be creating a set of keyboard layouts that are related to each other. In this sort of case, you may want to use the same dead key that is defined in the other keyboard layout. This command saves you the work of doing it all again by importing the dead key.

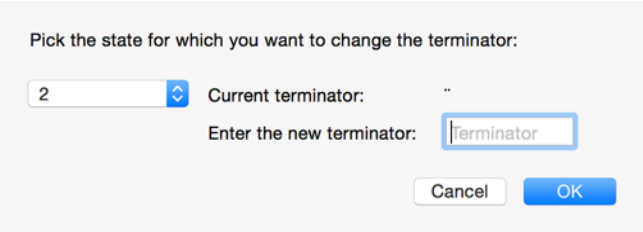
There is one constraint when you want to import a dead key. The issue is whether you have the same set of modifier combinations in both the keyboard layouts. If you do not have them exactly the same, then Ukelele will not allow you to import the dead key.

When you choose this command, Ukelele asks you to open the other keyboard layout by showing you a standard open file dialog. Ukelele then checks that the modifier key combinations are the same, and then asks you to identify the dead key state you want to import. It asks for the name of the dead key state rather than the actual dead key, and you choose it from the pop-up list. You then need to give the dead key state a name in the new keyboard layout, with Ukelele checking that the state name does not already exist in the keyboard layout into which you are importing the dead key.

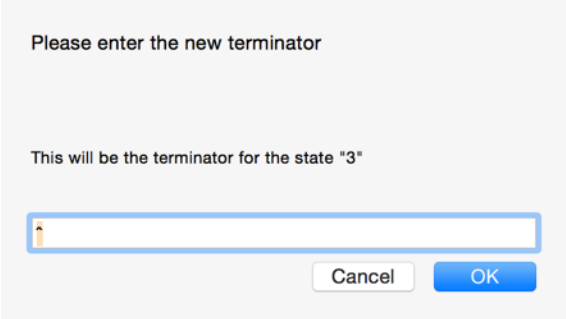
Change Terminator…

This command allows you to change the terminator for a dead key state, that is, the output that is generated when the key typed after the dead key produces no output in the dead key state.

If you are in state “none”, meaning that no dead key state is active, a dialog appears asking which dead key state you want to change the terminator for, and the new terminator. As you choose a dead key state, the dialog shows the current terminator.



If you are in any dead key state (i.e. not state “none”), then the dialog allows you to change the terminator for that state. The current terminator is in the text field and selected, for you to change it. The name of the dead key state is also listed in the dialog.



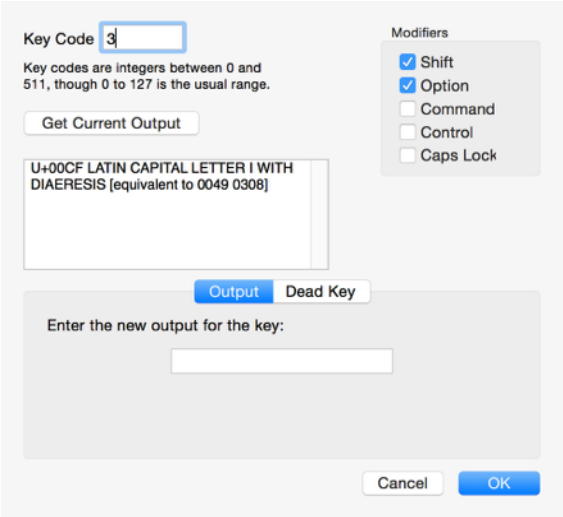
In either case, you can supply the new terminator by any method: typing it, entering the Unicode code point, pasting it, entering it via the character viewer, drag and drop, etc.

Select Key By Code…

Choosing this command brings up a dialog where you select a key by specifying its key code. Key codes must be in the range 0 to 511, but most key codes fall in the range 0 to 127. This is mostly useful for handling keys that are not on a given keyboard, e.g. numeric keypad keys on a laptop keyboard.

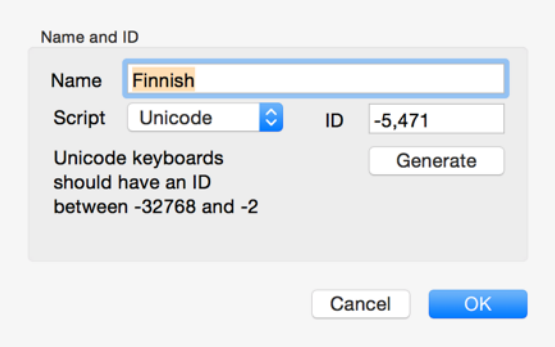
Edit Key…

When you choose this menu item, you get a dialog that allows you to specify a key by code, plus the modifiers that are to be associated with it. If a key is selected, then its code is filled in, and the current modifiers are set. Once you have entered a key code, you can click the Get Current Output button to see the output listed in the pane below it, in the same format as it is shown in the output pane of the Info Inspector. You can then enter new output for the key.



You can also turn a key into a dead key by selecting the Dead Key tab at the bottom of the dialog, and specifying the dead key state and (optionally) the terminator. As in creating a dead key, the terminator is only used if you are creating a new dead key state.

Unlink Key…

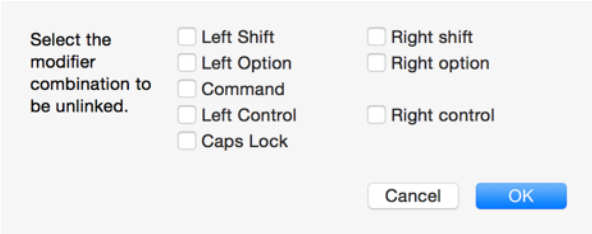


Some keys are “linked”, that is, changing the output of a key with one set of modifier keys affects the same key with a different set of modifier keys (or even in a different dead key state). The most common case of this is that the output of alphabetic keys with the shift key are linked to the same keys with the caps lock key down. Most of the time, this is what you want, but sometimes it isn’t, so Ukelele provides this command to unlink the key.

As usual, you can either select the key first and then choose this item (or click the icon in the toolbar), or choose it and then follow the prompt to select which key to unlink. In the normal state, the modifier keys that are pressed when the key is typed are those that are in effect. So, typing ⇪B makes the selected key B with caps lock down. However, if “sticky modifiers” has been turned on, then the modifier keys that are shown in the Ukelele window as down are the relevant modifiers.

If you are on the Modifiers tab rather than the Keyboard tab, this command has the same effect as Unlink Modifier Set...

Unlink Key By Code…



This is the same as Unlink Key..., but it brings up a dialog where you specify the key code. The modifiers that are relevant are those that are current when you choose the command.

Unlink Modifier Set…

This command is like the Unlink Key... command, but it applies to a whole set of keys, namely every key on the keyboard with a given set of modifier keys.

A dialog comes up which allows you to select the modifier combination. The checkbox should be checked for the modifier to be pressed, so the state when the dialog comes up is no modifiers pressed.

Note that the modifiers will look different if the keyboard layout has simplified modifiers, with only one column of modifiers, and no reference to left and right modifiers.

Swap Keys…

Sometimes you would like to swap two keys in a keyboard layout, like swapping Y and Z for the German standard as compared to the classic English QWERTY layout. You need to tell Ukelele which two keys to swap, and this can be done in a number of ways. Simplest is to select one of the keys before choosing this command, which only requires you to then choose the second key, which you do by typing or clicking it in the keyboard layout window. If you have not selected a key, you will have to specify both keys in turn by either typing or clicking. A prompt in the text area above the keyboard layout will guide you.

Swap Keys By ID…

This is similar to Swap Keys..., but you specify the keys by their key code. If you already have a selected key, then you only need to specify the second key’s code.

Set Keyboard Name and ID…

The name of a keyboard layout is what you will see in the input menu (the menu with the flag, usually), and in System Preferences when you are activating keyboard layouts. New keyboard layouts have a default name, and you should change it to something you will recognise.

All keyboard layouts have a numeric ID. In some ways, it shouldn't really matter what ID you give a particular keyboard layout, as the operating system will automatically renumber keyboard layouts that have the same ID. However, experience shows that this automatic renumbering causes some problems in using keyboard layouts, so it is better to have a unique ID if you can.

There is one important thing about a keyboard layout's ID. If it is to produce Unicode, it *must* have a negative ID. If you give a keyboard layout a positive ID, it will only work with one or more of the Mac "scripts", and will not be recognised as being Unicode.

That said, you can create a keyboard layout in Ukelele which isn't Unicode, but one of the scripts (Roman, Central European, Cyrillic, Japanese, Korean, Simplified Chinese or Traditional Chinese), and assign it a positive ID. The important point to remember is that you must only assign output that falls within the range of characters supported by the script that you set in the dialog. Otherwise, your users will end up with squares or question marks when they try to use your keyboard layout. Ukelele makes this easy by generating a random ID whenever you choose a particular script, and providing the Generate button to create a new ID whenever you like.

Set Keyboard Language…

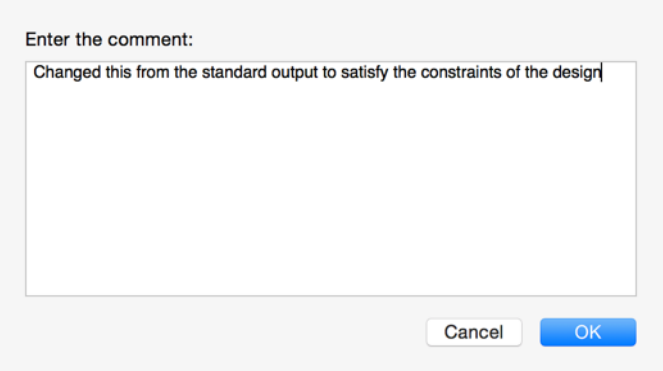
This command is only available when the frontmost window is a keyboard layout collection window, and a keyboard layout is selected. It is entirely the same as clicking the Language... button in the window.

Attach Icon…

This is the same as clicking the Icon... button in the keyboard layout collection window.

Add Comment…

It is possible to add a comment to a particular key with a given set of modifiers. You might do this for documentation purposes, such as indicating something you changed from somebody else's keyboard layout. This command is only enabled if you have a selected key. The modifiers current when you choose the command are those that are relevant. When you choose this command, a sheet comes up for you to enter your comment.



Add Special Key Output

Keyboard layouts from early versions of Ukelele, or from other sources, often don't include standard output for all the possible special keys. In particular, recent Apple keyboards have extra function keys, F16 to F19, which may not have any output defined in the keyboard layout. When you open a keyboard layout that is missing this output, you are asked whether you want to add the extra special key output. If you don't do it then, you can always do so with this menu item.

This should not change anything that you have done in your keyboard layout. It should only add standard output for special keys that do not yet have any output associated with them.

Change State Name…

For some uses, it is helpful to change the name of a dead key state. Particularly when you are starting from an automatically generated keyboard layout, such as one of those supplied with Ukelele, or one created by a tool like KeyLayoutMaker or Alex Eulenberg's web site, dead key states have names that are not helpful for remembering what they do. Also, unless you change the default, Ukelele assigns automatic names for the dead key states. When it comes to moving a dead key, or adding another dead key with the same state as an existing one, or working on multi-level dead keys, you need to know the name of the dead key state.

So, how do you find out the name of the dead key state that you want to change? The simplest way is to use the info inspector, if you know the dead key that you type to get the dead key state. Just move the mouse over the dead key (with whatever modifier keys need to be down), and you'll see the name of the dead key state in the info inspector. Another way is to double-click the dead key, and pick Enter State in the dialog that comes up, then look at the state stack for the name of the dead key state.

When you want to change the name of a dead key state to a name that you find more useful, you choose this menu item. It brings up a dialog that asks you to select the name of the state for which you want to change the name. You then get another dialog to enter the new name. Ukelele will not let you enter a name that has already been used.

Change Action Name…

This is similar to Change State Name..., but works with action names. It is extremely rare that anyone would want to do this, as you will never see an action name unless you look at the XML file. However, the option of changing an action name is present for completeness. Again, you get a dialog, choose the action you want, and then give it a new name in the next dialog.

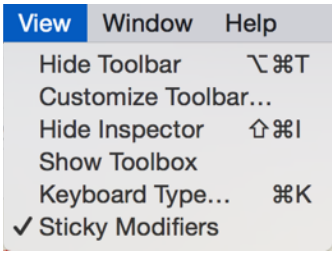
Remove Unused States

After you have worked with a keyboard layout for a while, if you have ever deleted a dead key, then the dead key state is actually still there, ready to be used again. If you don't want to use it again, it's just taking up space in the keyboard layout file, and a bit of memory in your computer. If you are confident that you don't want to reuse one of the deleted dead key states, this will remove them completely.

Remove Unused Actions

This is similar to Remove Unused States. Again, it does not change the functionality of the keyboard layout, but may remove some things that are no longer of use. An action is a collection of instructions that tell the operating system what to do given a key stroke and current dead key state, and is usually specific to a particular key and modifier combination. If it's no longer used, then you can't reuse it, unless you undo what caused it be no longer used.

8.2.6. View Menu



The View menu controls various things that are related to the display of the keyboard layout window and other floating windows.

Show/Hide Toolbar

This shows or hides the toolbar in the keyboard layout window. The keyboard shortcut is ⌥⌘T.

Customize Toolbar…

This is the same as clicking the Customize button on the toolbar, or choosing Customize... from the contextual menu you get from control-clicking (or right-clicking) on an empty spot in the toolbar. It opens a dialog that allows you to arrange the toolbar to your desired configuration.

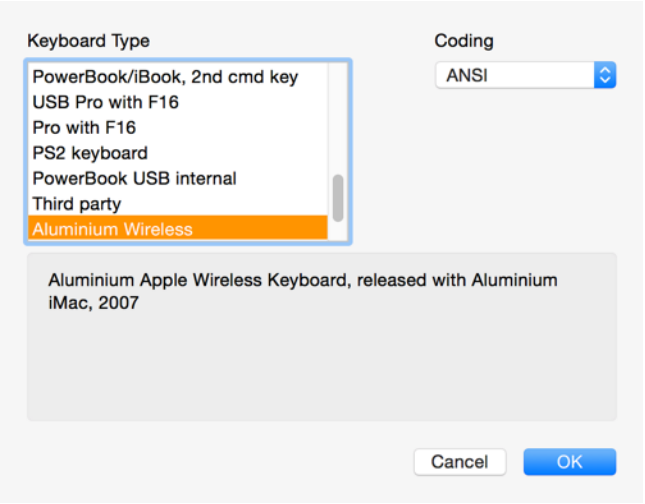
Show/Hide Inspector

This shows or hides the info inspector window. The keyboard shortcut is ⇧⌘I.

Show/Hide Toolbox

This shows or hides the toolbox window.

Keyboard Type…



The keyboard type is the physical arrangement of keys of a particular hardware keyboard, as represented in the Ukelele keyboard window. The arrangement of the keys was originally defined by a KCAP resource, which is defined by Apple. All of the keyboard types which were defined by KCAP resources have been made available for use in Ukelele.

When you select this item from the menu, you get the dialog box in the figure. The list on the left has all the distinct kinds of keyboards known to Ukelele. When you select one, a brief description is given in the box below.

In the top right of the dialog, there is a pop-up button which lists the type of keyboard, which could be ANSI, ISO or JIS. ANSI and ISO are virtually the same, with ISO having one extra key, often to the left of the Z key in a standard QWERTY keyboard layout, and some keys are moved. JIS keyboards have quite a different arrangement, and include some extra keys for composing Japanese characters.

The intent of having all the different keyboards available is to aid you in creating your keyboard layout. If you have a non-Apple keyboard, you can try various keyboard types to see if you can find one that is close to your hardware keyboard's layout. It's likely that you won't find a perfect fit, but you should be able to find something close enough to work with.

As you choose a keyboard type in the left hand pop-up button, the description below will change, and the other pop-up will change to reflect what possibilities there are for the keyboard type you've selected.

To emphasise a point here, this setting *only* affects how the keyboard layout is displayed by Ukelele. It has *no effect* on the actual keyboard layout, which will work with any hardware keyboard connected to a Mac.

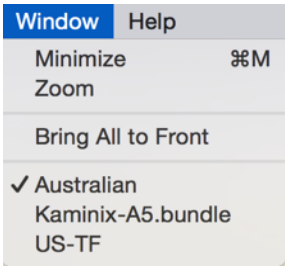
Sticky Modifiers

When this is active (a check mark shows in the menu), all the modifier keys behave in the same fashion as the caps lock key does. Pressing the modifier key once makes it set down, and pressing it again makes it set up. This is useful when setting the output of several keys with the same modifier key combination, for example.

Also when sticky modifiers are active, you can click on a modifier key in the Ukelele window, and that will toggle the modifier key up or down. The only caution is the caps lock key:

pressing the actual caps lock key takes precedence over the clicking. So, if you click the caps lock key to make it down, then press the caps lock key itself, you will see that it is still down. For all other modifier keys, a click and a key press are equivalent.

8.2.7. Window menu



This is a standard menu, allowing you to minimise, zoom, cycle through the windows, bring them all to the front, or choose a particular window to bring to the front.

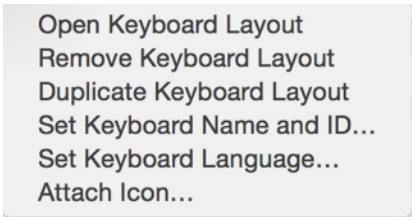
8.2.8. Help menu

Access to the Ukelele help book is through this menu. This manual is accessed through the menu as well.

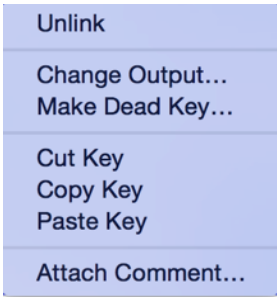
The other two items take you to the Ukelele web site at http://scripts.sil.org/ukelele, and to the Ukelele Users group at Google Groups.

8.3. Contextual Menus

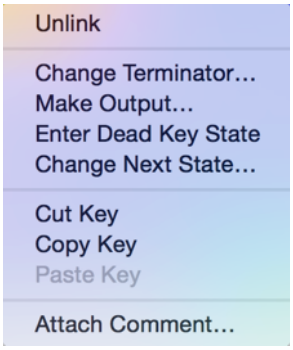
Many places in Ukelele have contextual menus, which are invoked by the standard methods of control-click or right click.



If you right click on a keyboard layout in a collection window, you get a contextual menu which allows you to open the keyboard layout in a keyboard layout window, remove it from the collection, duplicate it, set the keyboard layout's name, ID and language, or attach an icon to the keyboard layout.



If you right click a key in a keyboard layout window that is not a dead key, you get a contextual menu that allows you to unlink the key, change its output, make it into a dead key, cut, copy or paste the key, or attach a comment to the key. Pasting a key is only enabled if there is a key that has been cut or copied onto the pasteboard.



If you right click a dead key in a keyboard layout window, you get a contextual menu which allows you to unlink the key, changed the terminator of the dead key state, turn the dead key into an output key, enter the dead key state, change the dead key state the dead key triggers, cut, copy or paste a key, or attach a comment to the key. Pasting a key is only enabled if there is a key that has been cut or copied onto the pasteboard.

9. Troubleshooting

There shouldn't be much that can go wrong with a keyboard layout, but here is some advice for some situations.

9.1. My keyboard layout doesn't appear in the menu

There could be several reasons for this.

First, check that your keyboard layout is in the right location. It must be in a folder named Keyboard Layouts, inside one of the Library folders. Mostly, this will either be the Library folder in your own home folder, or in the Library folder at the top level of your hard disk. Please note that Library will be localised, so that it will be in the language that is set for your computer. The file name must have the extension .keylayout, which is automatically provided by Ukelele. Or if it is a keyboard layout collection, the collection must have the correct format and be in the correct folder. Collections created by Ukelele should have the correct format. Keyboard layouts installed by Ukelele will be in the right location.

If the keyboard layout is in the correct folder, have you logged out and logged in again? Older versions of the operating system only checks for new keyboard layouts when you log in. In fact, it first checks whether the Keyboard Layouts folder has changed since the last time you logged in, unless this is the first time since you restarted the computer. Only if the folder has had its contents changed does the system load new keyboard layouts. So, you may need to drag the keyboard layout file to the desktop, then drag it back to the Keyboard Layouts folder, and then log out and log back in. On at least OS X 10.10 (Yosemite), this should not be a factor.

Check again the relevant part of System Preferences. You're looking for the Keyboard (Language & Text on Mountain Lion) pane, and the Input Sources tab. Things are arranged slightly differently in different versions of Mac OS X, so you may have to hunt a little to find the correct place. Once you get there, you have either a list of active keyboard layouts, or a long list of all the input sources that are possible on your system, including keyboard layouts, input methods and the character viewer. These can be arranged in various orders by clicking on the header of each column. The name in the left hand column is where you need to look to find your keyboard layout. The name should be easy to find. If in doubt, check it in Ukelele by choosing the Set Keyboard Name... command from the Keyboard menu. When you find your keyboard layout, make sure that the checkbox is checked. Otherwise, it will not show up in the Input menu. On Yosemite, you need to click the + button and find your keyboard layout to make it active.

If you cannot find your keyboard layout in the Input Sources tab of Keyboard (Language & Text) preferences, that likely means that there is a problem with the keyboard layout, unfortunately. The problem will be shown by Apple's keyboard layout compiler, which is the program that loads the XML keyboard layout and turns it into the binary file that the operating system uses. The predecessor of the XML keyboard layout format was the 'uchr' resource. That's important to know because you will need to look for a line in your system's logs that has the string uchr in it. What you do is go to the Utilities folder in the Applications folder, and launch the

Console program. There should be a search field in the toolbar at the top. That's where you should enter uchr. Hopefully, it will show you a line that explains something about the problem.

If you can find a line which says something like "uchr XML compiler: when element specifying range must have numeric next state", then you have stumbled upon one of the keyboard layout compiler's bugs. What you need to do is to change some of your dead key state names. Find any name that is just a number — any dead key state whose name consists only of digits 0 to 9, and change its name so that it has at least one character that is not a digit. Doing this for all the dead key states whose names are numbers should solve this problem.

If you get a line in the log that has a different error, then you should seek some help. The best way is to compress your keyboard layout file (with something like StuffIt or zip), and send it to John Brownie at the email address in the Read Me file that comes with Ukelele. Also say what the error was, and the version of Ukelele and Mac OS X you were using.

9.2. I edited the keyboard layout, but the changes don't seem to work

You should always be editing a copy of the keyboard layout, not the installed copy. That means that you need to install the updated copy before it is in use. In an older version of OS X, you will need to log out and log in again after reinstalling the keyboard layout.

9.3. My keyboard layout doesn't work with application X

Again, there are several possible reasons for this. One is that the application may not support Unicode. These days, this generally means old applications. However, there have been some reports of problems with Microsoft Office 2008. These have yet to be confirmed. Microsoft Office 2011 appears to be better behaved, though there are occasional problems.

This problem might be a case of Mac OS X changing the keyboard layout without letting you know. The system tries to be helpful, but sometimes it is more aggravating than helpful. Often, when you switch applications, the system will switch the input source. Mostly, it will switch to one of the system's built-in keyboard layouts, usually the one that you had active when you logged in, or often just to the US keyboard layout. Make sure that your keyboard layout is actually selected in the application, by checking that there is a check mark beside it in the Input menu (the menu that usually has a flag at the top, towards the right of the menu bar). If you have a flag icon associated with your keyboard layout, it should be the one visible in the menu bar.

If you are still stuck, try checking the Ukelele Users group at Google Groups (http://groups.google.com/group/ukelele-users) or the Ukelele web site (http://scripts.sil.org/ukelele) and see if anyone else has a similar problem and has found a solution. If not, feel free to write a post on the Ukelele Users group explaining your problem. It's good to include the version of Mac OS X you are using, the program and its version that are causing the problem, and some details of what is happening.

9.4. I don't see the characters I expect when I type using my keyboard layout

If you see boxes or question marks instead of characters that you expected, the reason is most likely that you have your keyboard layout set to be non-Unicode. You should choose the Set Keyboard Name and ID... command from the Keyboard menu, then choose Unicode from the pop-up list, and ensure there is a negative number in the ID field.

If you are getting something else, then the first thing to do is to check that your keyboard layout is actually active. Mac OS X has had a nasty habit of changing the active keyboard layout without you knowing it, though the situation has improved with later releases of OS X. A

particularly disconcerting thing is that Safari will change the input source to a system keyboard layout when you are typing in a password field. Check the Input menu (the one with a flag) and see that your keyboard layout is active, that is, there is a check mark next to it in the menu. You can also set hot keys to switch between input sources (different keyboard layouts and input methods), though the default in 10.4 (Tiger) and 10.5 (Leopard) is the same as for Spotlight. Go to System Preferences, Keyboard & Mouse, Keyboard Shortcuts tab, and look for the Input Menu and Spotlight sections to set your desired combinations.

9.5. Some keys, such as arrow keys or enter, don't seem to work correctly, perhaps in only some applications

The most likely reason for this seems to be that the keyboard layout's ID is invalid for the given script code. Starting with Ukelele version 2.2, this conflict is detected when opening a keyboard layout file, and you are offered the option to correct this automatically.

If you do not want to fix this automatically, what you need to do is to select Set Keyboard Name and ID... from the Keyboard menu. Choose the appropriate script (usually Unicode) from the pop-up menu, and Ukelele will generate an appropriate ID for that range. Then go through the usual procedure of saving the keyboard layout and installing it. Hopefully it will work correctly after that.

Another reason is that in the next issue.

9.6. I wanted the forward delete key which is missing on my keyboard, so I made a keyboard layout that has a forward delete, but it doesn't work

This problem is not limited to the forward delete, but is related to all the special keys on the keyboard. Remember that special keys means those keys that normally don't produce visible output, but are used for control functions, and include delete, return, enter, escape, the arrow keys, home, end, page up, page down, forward delete, help, and the function keys (F1 to F19).

It appears that some applications, including several Apple applications, don't honour the output of special keys. Rather, they are built to recognise the key code and act accordingly. This means that you can neither turn a special key into a different key, nor turn an ordinary key into a special key. It will work for some applications, but not all.

9.7. The system keeps on changing away from my keyboard layout to a different one

This is a problem with different versions of the operating system. Solutions have been found for versions 10.2 (Jaguar) to 10.4 (Tiger), but these involve changing system files. For details, look at the Ukelele web site, and find the thread "input source change in Safari". The solution does not work on 10.5 (Leopard) or later, as Apple has changed the way that system keyboard layouts are provided.

Partial solutions are possible. One simple solution is to enable the hot keys or keyboard shortcuts for switching input sources. In versions 10.4 and 10.5, the standard keyboard shortcuts conflict with Spotlight, and may also conflict with LaunchBar, if you are using that. The preferences are set in different places in different versions. In 10.3 (Panther), you need to open System Preferences, go to the International (Keyboard on Mavericks, Language & Text on Snow Leopard, Lion or Mountain Lion) pane, choose the Input Menu (Input Sources on Snow Leopard or later) tab, and click on Options... to see the shortcuts. They are not customisable, but can only be switched on or off. Experiment also with the second option, "Try to match keyboard with text",

and see if it helps with the problem (try having it on and having it off). In 10.5 (Leopard), it is in the Keyboard & Mouse preference pane, in the Keyboard Shortcuts tab, down towards the bottom of the list. You can customise the keys here. This option does not appear to exist in 10.6 (Snow Leopard) or later.

9.8. The emacs control-key combinations don't work with my keyboard layout

In some ways, this is related to the general issue of control key combinations not working with custom keyboard layouts. However, there is one more problem. It appears that emacs uses its own system, separate from the standard key bindings. This means that, if you have edited your key bindings dictionary to reflect your keyboard layout, this won't apply to emacs. You either have to change the emacs key bindings, or overhaul the way your keyboard layout is organised.

The way that the control key is handled appears to be this. When a key stroke with the control key is sent to the system, if the option key is not down, or if the quote key (normally ^q) has not just been sent, the system reanalyses the key stroke. It does this by asking what character would be generated by the same key stroke without the control key. So, if you have a Dvorak keyboard layout, the 'q' key actually produces a straight single quote. So if you type ^q, the system would look for the key binding for control plus single quote.

What you may need to do is to create a new key bindings dictionary which handles your keyboard layout correctly. Directions on how to handle the key bindings dictionaries are at http://www.hcs.harvard.edu/~jrus/Site/cocoa-text.html. Basically, you want to substitute whatever your keyboard layout produces for the keys you want. For example, if your keyboard layout produces an aleph (א, Unicode U+05D0) for the 'a' key, then you would want to change the key binding from "^a" (meaning control-a) to "^\U5D0". If you are creating your own local key bindings dictionary, you would make a binding like "^\U5D0 = moveToBeginningOfParagraph:".

Please be aware that creating a keyboard layout that requires this sort of change will not be useful for anyone but you. Everyone who installs such a keyboard layout would need to make the same changes in key bindings, which is more than most people are willing to do, unless you can make it simple by supplying a ready-made DefaultKeyBinding.dict file. Even then, you will inconvenience people who have already modified their key bindings. So think carefully before going down this road!

10. Resources

This chapter contains links to various internet sites which give you access to some ready-made keyboard layouts, and some other sites with information that should be of interest to some users. Please check the terms of usage of keyboard layouts that you get from these sites. Some are freely available, others are shareware or commercial, so only distribute freely available ones, and get permission where needed to distribute modified versions of other people's keyboard layouts.

If you know of other sites that would be useful to others, please submit them to the author, and they will be included in future editions of this manual.

All links were verified on 4 June 2015.

URL	Site description
http://groups.google.com/group/ukelele-users	User group home page.
http://www.unicode.org	Details of the Unicode standard.
http://developer.apple.com/technotes/tn2002/tn2056.html	Apple's technical note describing the XML format for keyboard layout files.
http://scripts.sil.org/IWS-TOC	SIL's Non-Roman Script Initiative's book, "Implementing Writing Systems".
http://www.unibuc.ro/e/prof/paliga_v_s/soft-reso/	Sorin Paliga's home page (in English), with various keyboard layouts available for download.
http://www.evertype.com/celtscript/celt-keys.html	Celtic keyboard layouts from Everson Typography.
http://www.xenotypetech.com/	Assorted Unicode language kits from XenoType Technologies.
http://www.bekkoame.ne.jp/~n-iyanag/researchTools/asianextended.html	Nobumi Inayaga's Asian Extended keyboard layout.
http://www.ynlc.ca/languages/font/index.html	Yukon keyboard layout from the Yukon Native Language Centre.
http://apagreekkeys.org/	GreekKeys, a commercial package for polytonic Greek.
http://www.macupdate.com/app/mac/43412/vietnamese-keyboard-set	Vietnamese keyboard layouts by Gero Herrman.
http://ntresources.com/blog/?s=macintosh	Polytonic Greek keyboard layout by Rodney Decker.
http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=ipa-sil_keyboard	IPA keyboard for the SIL-IPA fonts.
http://www.alanwood.net/unicode/fonts_macosx.html	Unicode fonts and links to specific keyboard layouts.
http://www.linguistsoftware.com/lhebu.htm	Linguist Software's Laser Hebrew font and keyboard layout.
http://www.linguistsoftware.com/lgku.htm	Linguist Software's Laser Greek font and keyboard layout. They also have other keyboard layouts.
http://folk.uib.no/hnooh/mufi/keyboards/macOSXkeyboard.html	Medieval Unicode Font Initiative keyboard layout.
http://www.thlib.org/tools/	Nepali fonts and keyboard layouts from the Tibetan & Himalayan Digital Library (you will need to navigate to Fonts & Related Issues, then to Nepali Fonts).

URL	Site description
http://www.mellel.com/downloadother#keyboardlayouts	Various keyboard layouts linked to by RedleX, makers of Mellel, a Unicode-savvy word processor.
http://madanpuraskar.org/?page_id=157	Nepali Unicode Romanized keyboard layout from Madan Puraskar Pustakalaya.
http://www.ukij.org/mac/	Links to Uyghur software, including keyboard layouts.
http://www.aatseel.org/macintosh_cyrillic	Cyrillic fonts and keyboard layouts.
http://arkku.com/dvorak/#chapter3_1	Dvorak layout for Finnish by Kimmo Kulovesi.
http://ebmp.org/p_dwnlds.php	Unicode keyboard layouts from the Early Buddhist Manuscripts Project (you want EasyUnicode).
http://www.unirioja.es/cu/jvarona/TeXkeylayout.html	Keyboard layouts for typing TeX/LaTeX by Juan L. Varona.
http://m10lmac.blogspot.com/	Tom Gewecke's blog page, with reference to many keyboard layouts.
http://sites.google.com/site/macmalayalam/	Fonts and keyboard layout for typing in Malayalam.
http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=MacHGTransUniKey	Keyboard layout for transliterated Greek and Hebrew, developed by Joan Wardell.
http://www.hcs.harvard.edu/~jrus/Site/cocoa-text.html	Information on key bindings and the Cocoa text system.
http://belkadan.com/blog/2012/04/Keyboard-Adventures/	Information about how the "intended language" system works with OS X and how to do more with the "press and hold" system.
http://en.wikipedia.org/wiki/IETF_language_tag	Wikipedia article about language tags used for the "intended language" system.
http://apple.stackexchange.com/questions/20505/add-characters-to-the-press-and-hold-character-picker-in-os-x-lion/44928#44928	How to customise the options for "press and hold". Note that this is not something you can distribute with keyboard layouts created by Ukelele, but something that you can do to your own system.

A. Languages supported by OS X

Apple introduced "press and hold" in OS X 10.7 (Lion), which allows you to press a key and hold it down for a short period until a pop-up appears to offer variants of that character for easy entry of diacritics. However, the options are determined by the language associated with the keyboard layout in its bundle. The following table lists the languages supported as of OS X 10.10.2.

Language	Code	Notes
Arabic	ar	Arabic scrip
Bulgarian	bg	Cyrillic script
Catalan	ca	Also used for Valencian
Cherokee	chr	
Chinese	zh	zh-Hans (Simplified), zh-Hant (Traditional). Also variants Wubihua for Simplified and Cangjie, Pinyin, Wubihua and Zhuyin for Traditional.
Croatian	hr	Latin script
Czech	cs	
Danish	da	
Dutch	nl	Also nl-BE (Belgium), i.e. Flemish
English	en	Also en-AU (Australia), en-CA (Canada), en-GB (Great Britain), en-NZ (New Zealand), en-US (USA)
Estonian	et	
Finnish	fi	
French	fr	Also fr-CA (Canada), fr-CH (Switzerland), fr-FR (France)
German	de	Also de-CH (Switzerland)
Greek	el	Greek script
Hebrew	he	Hebrew script
Hungarian	hu	
Icelandic	is	
Indonesian	id	
Italian	it	
Japanese	ja	Does not specify script
Korean	ko	Korean script
Latvian	lv	
Lithuanian	lt	
Macedonian	mk	
Malay	ms	
Norwegian	nb	Bokmål
Polish	pl	
Portuguese	pt	Also pt-BR (Brazil), pt-PT (Portugal)
Romanian	ro	Latin script. Also used for Moldovan, Moldavian.
Russian	ru	Cyrillic script

Language	Code	Notes
Serbian	sr	sr-Cyrl (Cyrillic script), sr-Latn (Latin script)
Slovak	sk	
Spanish	es	Castilian
Swedish	sv	
Thai	th	Thai script
Tibetan	bo	
Turkish	tr	
Ukrainin	uk	Cyrillic script
Vietnamese	vi	

The support for each language is somewhat varied, with some offering rich sets of variants, while others have few if any. If you are interested to find out what each language supports, then look in the files in /System/Library/Input Methods/PressAndHold.app/Contents/Resources. These are property lists, which can be examined simply with TextEdit. A sample section for a couple of languages are below:

```
<key>Roman-Accent-a</key>
<dict>
    <key>Direction</key>
    <string>right</string>
    <key>Keycaps</key>
    <string>a à á â ä æ ã å ā</string>
    <key>Strings</key>
    <string>a à á â ä æ ã å ā</string>
</dict>
```

This is from the English set, and shows the variations offered when the “a” key is held down.

```
<key>Roman-Accent-a</key>
<dict>
    <key>Direction</key>
    <string>right</string>
    <key>Keycaps</key>
    <string>à â ª a æ á ä ã å ā</string>
    <key>Strings</key>
    <string>à â ª a æ á ä ã å ā</string>
</dict>
```

This is from the French set, and also shows variants for the “a” key. Note that the two sets are different.