



# **Ukelele**

## **Unicode Keyboard Layout Editor**

**Version 2.2**



# Contents

1. Introduction.....	1
1.1. What is Ukelele? .....	1
1.1.1. What Ukelele can do .....	1
1.1.2. What Ukelele can't do.....	1
1.1.3. The name Ukelele .....	3
1.1.4. Help .....	3
2. Installation.....	5
3. Quick Start.....	7
3.1. Making changes to a keyboard layout.....	7
3.1.1. Drag and drop .....	8
3.1.2. Double-click.....	8
3.1.3. Entering code points.....	9
3.2. Creating a new keyboard layout.....	9
3.2.1. Empty keyboard layout .....	9
3.2.2. Based on an existing layout .....	9
3.2.3. Capture the current keyboard input source .....	9
3.3. Installing and using a keyboard layout .....	9
3.4. Tutorial.....	10
4. Keyboard Layouts.....	11
4.1. How a keyboard layout works.....	11
4.2. Unicode and other scripts .....	11
4.3. Input methods .....	12
4.4. Dead keys .....	12
4.5. Using a keyboard layout .....	13
4.6. Support for “press and hold” .....	14
5. Creating and Editing a Keyboard Layout.....	15
5.1. Designing a keyboard layout.....	15
5.1.1. Who will be the users of the keyboard layout? .....	15
5.1.2. What is distinctive about the keyboard layout?.....	15
5.1.3. What Unicode characters will the keyboard layout produce?.....	15
5.1.4. Are dead keys a useful part of the keyboard layout?.....	15
5.2. Choosing a starting point .....	16
5.3. Creating a keyboard layout in Ukelele .....	16
5.3.1. Starting from an empty keyboard layout .....	16
5.3.2. Starting from an existing keyboard layout .....	16
5.3.3. Starting from a non-XML keyboard layout .....	17
5.4. Working in the Ukelele window .....	17
5.5. Editing key output .....	18
5.5.1. Drag and drop .....	19
5.5.2. Double-click.....	20
5.6. Creating and editing dead keys .....	20
5.6.1. What use is a dead key?.....	21
5.6.2. Principles for designing dead keys .....	21
5.6.3. Creating a new dead key .....	21
5.6.4. Editing an existing dead key.....	23

5.6.5. Moving a dead key.....	23
5.6.6. Multi-level dead keys.....	24
5.6.7. Terminators.....	25
5.6.8. Importing dead keys .....	25
5.6.9. Deleting a dead key .....	26
5.7. Managing modifier key combinations .....	26
5.7.1. What is a modifier key?.....	26
5.7.2. What is possible vs. what is useful .....	27
5.7.3. Creating new modifier key combinations .....	28
5.8. Installing a keyboard layout .....	32
5.8.1. Choosing the folder for installation .....	33
5.8.2. Keyboard layout bundles .....	33
6. Miscellaneous tasks .....	35
6.1. Comments .....	35
6.2. Changing dead key state and action names .....	36
6.3. Removing unused dead key states and actions.....	36
6.4. Adding special key output.....	36
6.5. Searching for an output string .....	37
7. Reference.....	39
7.1. Windows.....	39
7.1.1. Keyboard layout window .....	39
7.1.2. Modifier combinations drawer .....	42
7.1.3. Comments editor.....	43
7.1.4. The info inspector .....	44
7.1.5. Toolbox .....	44
7.2. Menus .....	45
7.2.1. Ukelele menu .....	45
7.2.2. File menu .....	48
7.2.3. Edit menu .....	50
7.2.4. Keyboard menu.....	51
7.2.5. View Menu .....	58
7.2.6. Window menu.....	62
7.2.7. Help menu .....	62
8. Troubleshooting.....	63
8.1. My keyboard layout doesn't appear in the menu.....	63
8.2. I edited the keyboard layout, but the changes don't seem to work .....	64
8.3. My keyboard layout doesn't work with application X.....	64
8.4. I don't see the characters I expect when I type using my keyboard layout.....	64
8.5. Some keys, such as arrow keys or enter, don't seem to work correctly, perhaps in only some applications .....	65
8.6. 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 .....	65
8.7. The system keeps on changing away from my keyboard layout to a different one	65
8.8. The emacs control-key combinations don't work with my keyboard layout .....	66
9. Resources.....	67
10. Languages supported by OS X.....	71

# I. Introduction

---

## I.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.

Ukelele is copyright © John Brownie, SIL, 2003–12, and is provided under SIL's free-ware licence (<http://www.sil.org/computing/catalog/freeware.html>). Its web site is <http://scripts.sil.org/ukelele>. Ukelele 2.2 is compatible with Mac OS X 10.4 (Tiger) and later, and is a Universal Binary. Older versions are compatible with Mac OS X 10.2 (Jaguar) and later.

### I.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, 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 option-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.

### I.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 DoubleCommand (<http://doublecommand.sourceforge.net/>) or KeyRemap4MacBook (<http://mac.pqrs.org/macosex/>)

### 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 Ъ, a Greek π, an Arabic ص, a Hebrew ל, a Gurumkhi ௃, a Japanese ク, a Korean 뽕, a Chinese 嘶, or even a letter from an archaic script like Ugaritic. It can also be a symbol, such as punctuation marks (-, ., —, & ;), phonetic symbols like d̥, 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, 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.

[keyremap4macbook/](#)), 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.

### **1.1.3. 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.

### **1.1.4. 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.ukелеle.plist`, which contains Ukelele's preferences. Inside a folder called Ukelele, inside the preferences folder, there is a file called `ColourThemes.plist`, which is an XML file that defines the colour themes available in Ukelele.

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 folders, each containing a set of keyboard layouts and their associated flag icons. These are XML versions of the keyboard layouts available on Mac OS X 10.4, and are intended as starting points for creating new keyboard layouts. The Logitech Keyboard Layouts folder 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. 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 Finnish keyboard layout, found in the Roman sub-folder of the System Keyboards folder provided with Ukelele.

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, control or fn) 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 Finn-

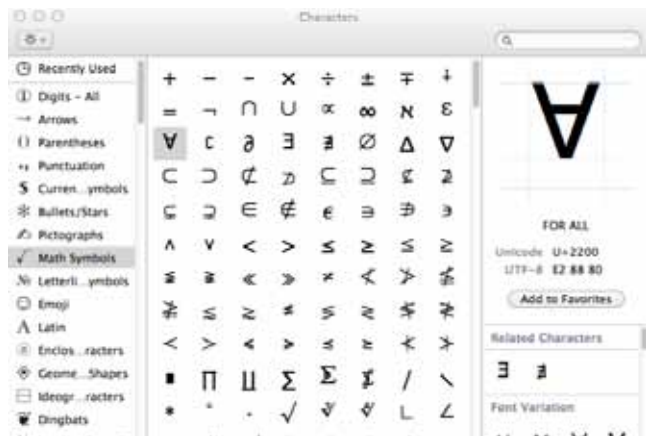


ish keyboard layout, the mathematical symbol “for all”,  $\forall$ . There are several ways to do this:

### 3.1.1. Drag and drop

If the Character Viewer (or Palette) 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 Language & Text (International on 10.5 or earlier) 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  $\forall$  character. Now, hold down shift and option, so that you can see the “ on the 2 key. Select the  $\forall$  character in the character palette, and drag it onto the 2 key. 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 dialog 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 OK, and you're done.

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



### 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 `&#x2200;` or `&#8704;`. 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 an empty 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. Empty keyboard layout

To create an empty keyboard layout, choose New from the File menu, or press  $\text{⌘-N}$ .

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. If you choose New Based On... from the File menu, or press  $\text{⌘-N}$ , 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. Ukelele will open an unsaved copy of that 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

Once you have a complete keyboard layout, you need to install it. There are three places that you can install a keyboard layout. The first is within the Keyboard Layouts sub-folder of the Library folder in your home folder. This can be created if it doesn't already ex-

ist. If you install it there, only you will be able to use the keyboard layout. Other users on the same computer will not have access to it.

To allow all users to use the keyboard layout, install it into the Keyboard Layouts sub-folder of the Library folder at the top level of your start-up disk. Again, create the folder if it doesn't already exist.

If you are on a server, and you want to make the keyboard layout to all users on the network, put the keyboard layout in the Keyboard Layouts sub-folder of the Library folder in the Network folder at the top level.

Most importantly, after you have installed a keyboard layout, log out and log in again, or restart the computer.

After installing the keyboard layout and logging out and logging in again, open the Language & Text pane of System Preferences (International on Mac OS X 10.5 (Leopard) and earlier). On the Input Sources (Input Menu in 10.5 or earlier) tab, your new keyboard layout should be listed — you will likely need to scroll the list down to find it. Click the check box next to it to make it active. You should then see it in the input menu. Select it, and you should now be able to type with your keyboard layout, as long as you are in an application that handles Unicode.

### 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.

## 4. 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.

### 4.1. 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 “\$”.

### 4.2. 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.

### **4.3. 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 (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.

### **4.4. 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, â, and ã, 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 option-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.

## 4.5. 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. 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 Language & Text (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 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.

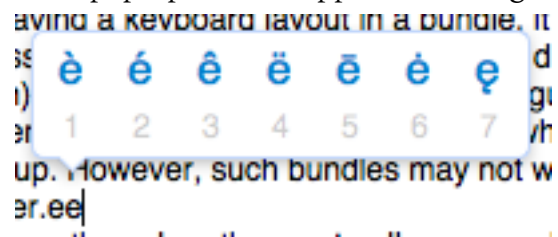
#### 4.6. Support for “press and hold”

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 2.2 introduces support for language marking in keyboard layout bundles. There is a new menu item, “Set Keyboard Language...” which allows you to choose the language associated with the keyboard layout. 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 5.8.2 for more details.



## 5. 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.

### 5.1. 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.

#### 5.1.1. 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.

#### 5.1.2. 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?

#### 5.1.3. 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.

#### 5.1.4. 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?

## 5.2. 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>).

## 5.3. 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. So, open Ukelele, and either choose File > New (⌘N) to start with a blank or empty keyboard layout, or File > New Based On... (⇧⌘N). This brings up a dialog box which allows you to choose an existing keyboard layout as your starting point.

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 section on Resources at the end of this manual for some pointers to available keyboard layouts.

### 5.3.1. 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.

### 5.3.2. 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, choose File > New Based On... (⇧⌘N). This will create a new keyboard which is a copy of whichever keyboard layout you open from this dialog.

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 folders, 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



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 “Set Keyboard Type...”, you are presented with a dialog that allows you to choose what kind of keyboard the display should show. The popup menu on the left lists 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.

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, control or fn 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 it will change state, just as if you’d pressed the corresponding key.

## 5.5. 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.

### 5.5.1. 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 (called Character Palette in 10.5 (Leopard) and earlier). You will find it in the input menu, which usually looks like a flag in the menu bar. If you don't have an input menu, that probably means that it hasn't been enabled, or that you only have one keyboard layout enabled. Either way, you need to open the Language & Text (or International on Leopard or earlier) pane of System Preferences, and switch to the Input Sources (Input Menu on Leopard or earlier) tab. At the bottom of the tab, there is a check box, "Show input menu in menu bar". Check that, and you should see the menu turn up.

☒ Show input menu in menu bar

If you haven't got the Character Viewer available, you need to go to the same place in System Preferences. You need to find the Character Viewer (or Palette) in the list, 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 10.7 version.

You use the Character Viewer with drag and drop by 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.

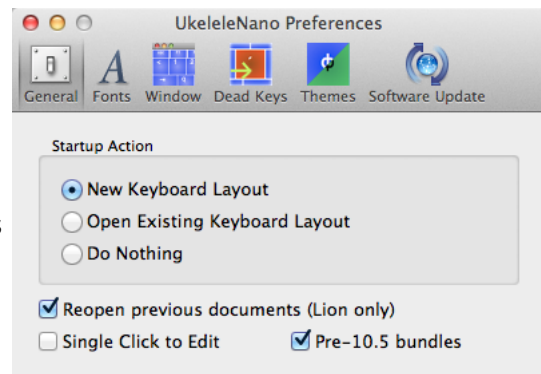
You don't have to use the Character Viewer, though. You can select text in a Unicode-aware application and drag that onto the key that you want to change. So, for example, if someone has put together a document which has all the characters you want, just open it in a program like TextEdit, Nisus Writer (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.

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.

### 5.5.2. 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.



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 (Character Palette in 10.5 or earlier), 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 “&#x00A1;” without the quotation marks. To enter it in decimal, you use “&#161;” 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 “&#xa1;” 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 “&#x1D110;&#x1D12B;” specifies “”. If you put a space, it is considered part of the output, so “&#x1D110; &#x1D12B;” includes the space, so it is “ ”

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.

## 5.6. 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.

### 5.6.1. 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.

### 5.6.2. 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.

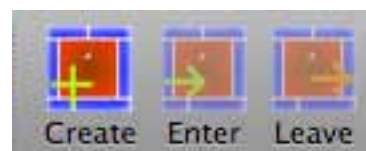
Consistency is very important. If a dead key produces a particular modification, such as adding a diacritical mark to a letter, then it should do the same for other letters, when this makes sense. A dead key that produces one diacritic for some letters, and a different one for other letters, will be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

### 5.6.3. Creating a new dead key

Once you have decided on a dead key, you create it by choosing “Edit Dead Key...” from the Keyboard menu, or click on the “Create” button in the toolbar. In the dialog that appears, click “New”. At this point, Ukelele is waiting for you to either type the dead key, or to click the dead key in the window, and the status bar will indicate what you should be doing next. When clicking, the modifiers used are whatever are currently down when you click, 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. There are two options, either creating a new dead key state with a name given in the dialog, or making the dead key enter an existing dead key state, which you choose from the pop-up menu. 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.

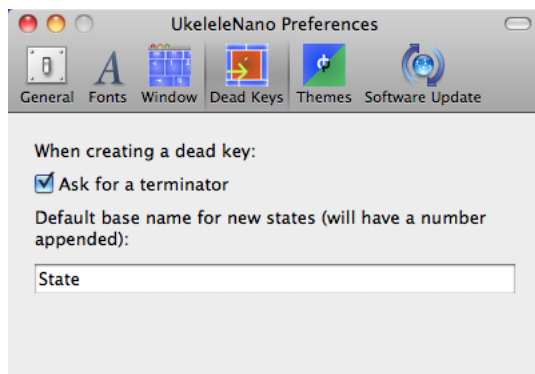
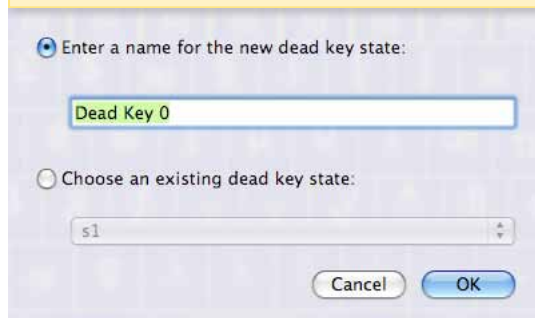
Press or click the new dead key

## 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 “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.



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 is given when creating the dead key, you can change the name later by choosing “Change State Name...” from the Keyboard menu.

If you give the name of a dead key state that already exists, then Ukelele will tell you this, and stop creating the new dead key.

What happens next depends on the settings you have in preferences. There is an option, that is turned off by default, for Ukelele to ask for the terminator for the new dead key.

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 option-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 5.6.7 on terminators for more information.

Once you have typed your dead key, and provided whatever further information you need (state name or terminator), most of the keys will appear blank. What you see is the output of each key in the new dead key state.

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.

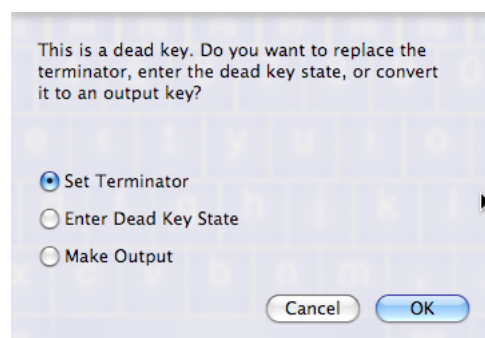
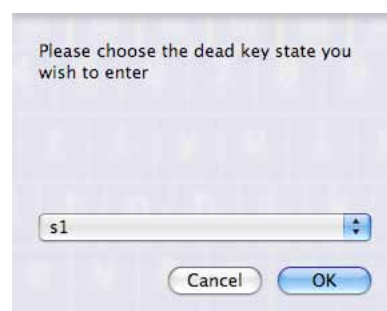
#### 5.6.4. Editing an existing dead key

There are times when you want to modify an existing dead key. For example, you may want to add another accentuated 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 5.6.6 for more on this).

The procedure for editing an existing dead key is much the same as for creating a new dead key. Select “Edit Dead Key...” 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 Dead Key State radio button to edit the dead key state.

Alternatively, 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.



#### 5.6.5. 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  $\text{⌘} \text{⇧} \text{-j}$  instead of  $\text{⌘} \text{-ñ}$ .

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 ( $\text{\`}$ ), and move the mouse over the key that produces  $\text{\~n}$  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 need to choose “Create Dead Key...” again, or click the “Create” button on the toolbar. Then press your new dead key (such as  $\text{\`}\hat{\text{u}}\text{-j}$  in the example above). When the dialog box appears to ask for a name for the dead key state, choose the “Choose an existing dead key state” radio button, 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 5.6.9.

### 5.6.6. 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.

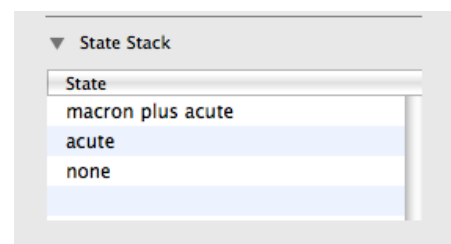
An example would be when you are creating a keyboard layout that is used for a language where you need to specify both length, by means of a macron over a vowel, and tone by means of an acute accent over the vowel. With only single dead keys, you would need to have separate dead keys for macron, acute 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  $\text{\`}\text{-a}$  as the dead key for a macron, and  $\text{\`}\text{-e}$  for an acute accent. We first create the  $\text{\`}\text{-a}$  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). Choose Edit Dead Key... from the Keyboard menu or click the “Create” button on the toolbar. Next, type  $\text{\`}\text{-a}$ . In the dialog, choose the “Enter a name for the new dead key state” radio button, and enter “macron” as the name. If you are asked 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 choosing Edit Dead Key... again or clicking the “Create” button on the toolbar, and typing the dead key for the acute accent,  $\text{\`}\text{-e}$ . 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, choose Cre-



ate Dead Key... or click the “Create” button on the toolbar, and type the dead key,  $\text{U+00B4}$ -e. Make the name of the dead key state “acute”, and the terminator (if you want it) the acute accent,  $\text{U+00B4}$ . Then add the output for the acute accent state.

To complete the work, choose Create Dead Key... again or click the “Create” button on the toolbar, and type the dead key for macron,  $\text{U+00B4}$ -a. When the dialog comes up for the name of the dead key state, choose the “Choose an existing dead key state” radio button, and 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.

### 5.6.7. 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, if the appropriate option is set in the preferences. 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, or double-click 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.

### 5.6.8. 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.

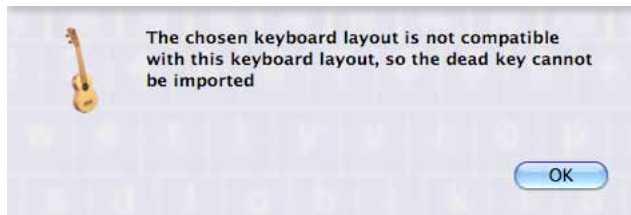
Importing a dead key means copying a dead key state, meaning all the output associated with that state. You can choose the key that will become the dead key in the keyboard that you are editing, as well as the name of the dead key state.

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



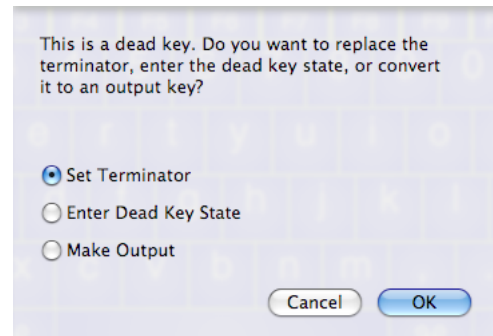


from the pop-up menu, and click OK. The next dialog asks you to name the dead key state in the keyboard layout you are editing, suggesting the name from the source keyboard layout. As usual, if the name is already taken, you will be offered the chance to give a new name.

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.

### 5.6.9. 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 a new terminator, entering the dead key state, or making the key an output key and specifying the new output.



## 5.7. 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.

### 5.7.1. 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.

### 5.7.2. 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 that very few keyboards will actually supply different codes for the left and right modifier keys. In other words, in almost every case 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.

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 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.

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  $\text{⌘}^{\text{⌥}}$ ,  $\text{⌘}^{\text{⌥}^{\text{⌥}}}$  or  $\text{⌘}^{\text{⌥}^{\text{⌥}^{\text{⌥}}}}$ , 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  $\text{⌘}^{\text{⌥}}\text{-q}$  means that the following key stroke will be handled without interpreting it as a command. So, if your keyboard layout has  $\text{⌘}^{\text{⌥}}\text{-f}$  set to produce f (U+0192, Latin letter f with hook), typing  $\text{⌘}^{\text{⌥}}\text{-q}^{\text{⌥}}\text{-f}$  should work. This depends on your keyboard pro-

### Control key combinations and key bindings.

Key bindings are low-level operations that make the system perform certain operations when key combinations are pressed, particularly the control key. They are controlled by the `StandardKeyBinding.dict` and `DefaultKeyBinding.dict` files. The former is a system file, so you must be extremely careful if you want to modify it, as it could make your system unusable if you get it wrong.

You can find a large amount of information about all this at <http://www.hcs.harvard.edu/~jrus/Site/cocoa-text.html>. It tells you the mechanics of the key binding system, locations of files, what they should look like, and what they can do. It is detailed, but worth skimming to get a feel for what you can do.

The important point is to note that the user's `DefaultKeyBinding.dict` can replace behaviour defined in the system's `StandardKeyBinding.dict`, but cannot remove behaviour defined there. In particular, there is one entry in the system's file that makes any control key combination not specifically set into a null operation. That entry is just "^" for the key, and "noop:" for the value. Removing that entry may help you to get what you want working.

For editing the key binding dictionaries, you can use the format that Jacob Rus mentions, or you can use Property List Editor, which is installed with the Developer Tools, or with a utility like KeyBindingsEditor (Mac OS X 10.4 or later, <http://www.cocoabits.com/KeyBindingsEditor/>). It is best to restart the computer after changing this file, to ensure that your changed version gets used.

ducing 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.

### 5.7.3. 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 opening the modifiers drawer, by choosing Show Modifiers from the View menu (or using the keyboard



shortcut ⌘-M), or clicking the “Modifiers” button on the toolbar. You also edit the complete set of modifier key combinations from the modifiers drawer.

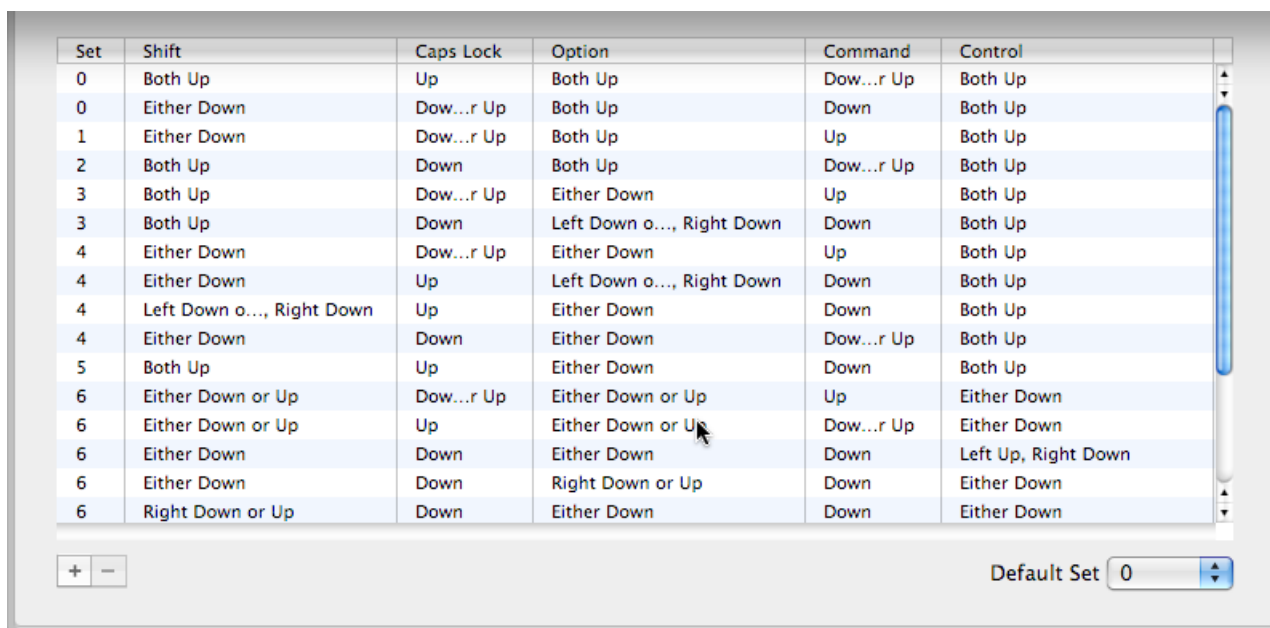
### Understanding the modifiers drawer

The modifiers drawer 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



Set	Shift	Caps Lock	Option	Command	Control
0	Both Up	Up	Both Up	Down...r Up	Both Up
0	Either Down	Down...r Up	Both Up	Down	Both Up
1	Either Down	Down...r Up	Both Up	Up	Both Up
2	Both Up	Down	Both Up	Down...r Up	Both Up
3	Both Up	Down...r Up	Either Down	Up	Both Up
3	Both Up	Down	Left Down o..., Right Down	Down	Both Up
4	Either Down	Down...r Up	Either Down	Up	Both Up
4	Either Down	Up	Left Down o..., Right Down	Down	Both Up
4	Left Down o..., Right Down	Up	Either Down	Down	Both Up
4	Either Down	Down	Either Down	Down...r Up	Both Up
5	Both Up	Up	Either Down	Down	Both Up
6	Either Down or Up	Down...r Up	Either Down or Up	Up	Either Down
6	Either Down or Up	Up	Either Down or Up	Down...r Up	Either Down
6	Either Down	Down	Either Down	Down	Left Up, Right Down
6	Either Down	Down	Right Down or Up	Down	Either Down
6	Right Down or Up	Down	Either Down	Down	Either Down

At the bottom of the drawer, there are buttons for adding (+) and removing (-) sets, and a dropdown menu for the Default Set, currently set to 0.

and left keys, so only the first three are generally relevant.

In each row, there is an entry for each of the modifier keys. For those with no left and right, that is caps lock and command, each entry may be Up, Down, or Down or Up. These should be obvious as to their meaning, but the last option means that it doesn't matter what the state of that particular modifier key is: it matches whether it is down or up.

One important thing to note first is the first column, labelled Set. 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:

	Shift	Caps Lock	Option	Command	Control
0	Both Up	Up	Both Up	Up	Both Up
1	Either Down	Down or Up	Both Up	Up	Both Up
2	Both Up	Up	Either Down	Up	Both Up
3	Either Down or Up	Down	Both Up	Up	Both Up
4	Either Down	Down or Up	Either Down	Up	Both 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 drawer. 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:

	Shift	Caps Lock	Option	Command	Control
0	Both Up	Up	Both Up	Up	Both Up
0	Either Down or Up	Down or Up	Both Up	Down	Both Up
1	Either Down	Down or Up	Both Up	Up	Both Up
2	Both Up	Down	Both Up	Up	Both Up
3	Both Up	Up	Either Down	Up	Both Up
4	Either Down	Down or Up	Either Down	Down or Up	Both Up
5	Both Up	Down	Either Down	Up	Both Up
6	Both Up	Down or Up	Either Down	Down	Both Up
7	Left Down or Up	Down or Up	Left Down or Up	Down or Up	Left Down, Right Up
7	Left Down or Up, Right Down	Down or Up	Left Down or Up	Down or Up	Left Down, Right Up
7	Left Down or Up	Down or Up	Left Down or Up, Right Down	Down or Up	Left Down, Right Up

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 shift, option and 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.

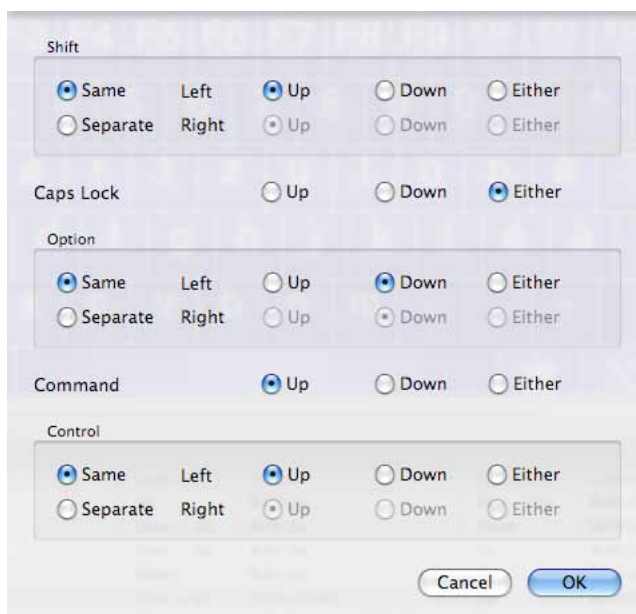
A second important thing to see in this set is that there are two rows with 0 in the first

(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).

### 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. It should be self-explanatory, but a quick tour may help.

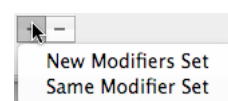
There is a separate section for each of the five kinds of modifier key. For those that have left and right keys, that is, shift, option and control, there is a radio button for whether they are considered the same or separate. Once again, you would normally have “Same” selected, as most keyboards consider 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.

### Adding a new modifier key combination

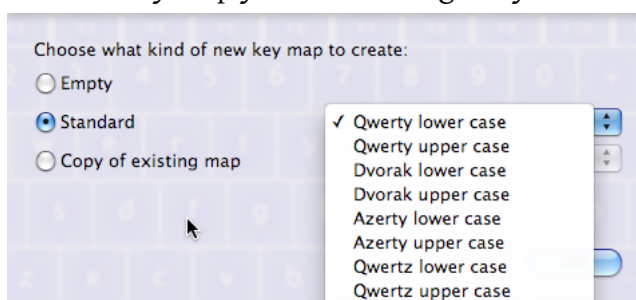
When you want to add a new modifier key combination, you have to make a decision before you start: Is it going to be a completely new modifier key combination, or is it going to be an alternate for an existing combination? In other words, is it going to appear with a new index number, or is going to be a row with the same index as an existing row? Either way, you add a modifier combination by clicking on the “+” button at the bottom of the drawer.



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 “Same Modifier Set” from the menu.

If you want a completely new modifier key combination, choose “New Modifiers Set” from the menu. You will then be asked what kind of key map you want. This gives you a set of outputs for all the keys on the keyboard. You can choose a standard set (see the sidebar What are the standard key maps?) from the popup menu, or make it a copy of a set that you already have for an existing modifier combination.

In any case, the same dialog comes up as is used when editing a modifier key



What are the standard key maps?

There are nine standard key maps provided. Each provides standard output for the “control keys”, that is, the keys that aren’t normal alphabetic, numeric or symbol keys, such as return, enter, backspace, escape, the arrow keys, page up, page down, home, end, help, and the function keys. Otherwise, only the alphabetic keys (a-z) are provided, not any of the digits or symbols.

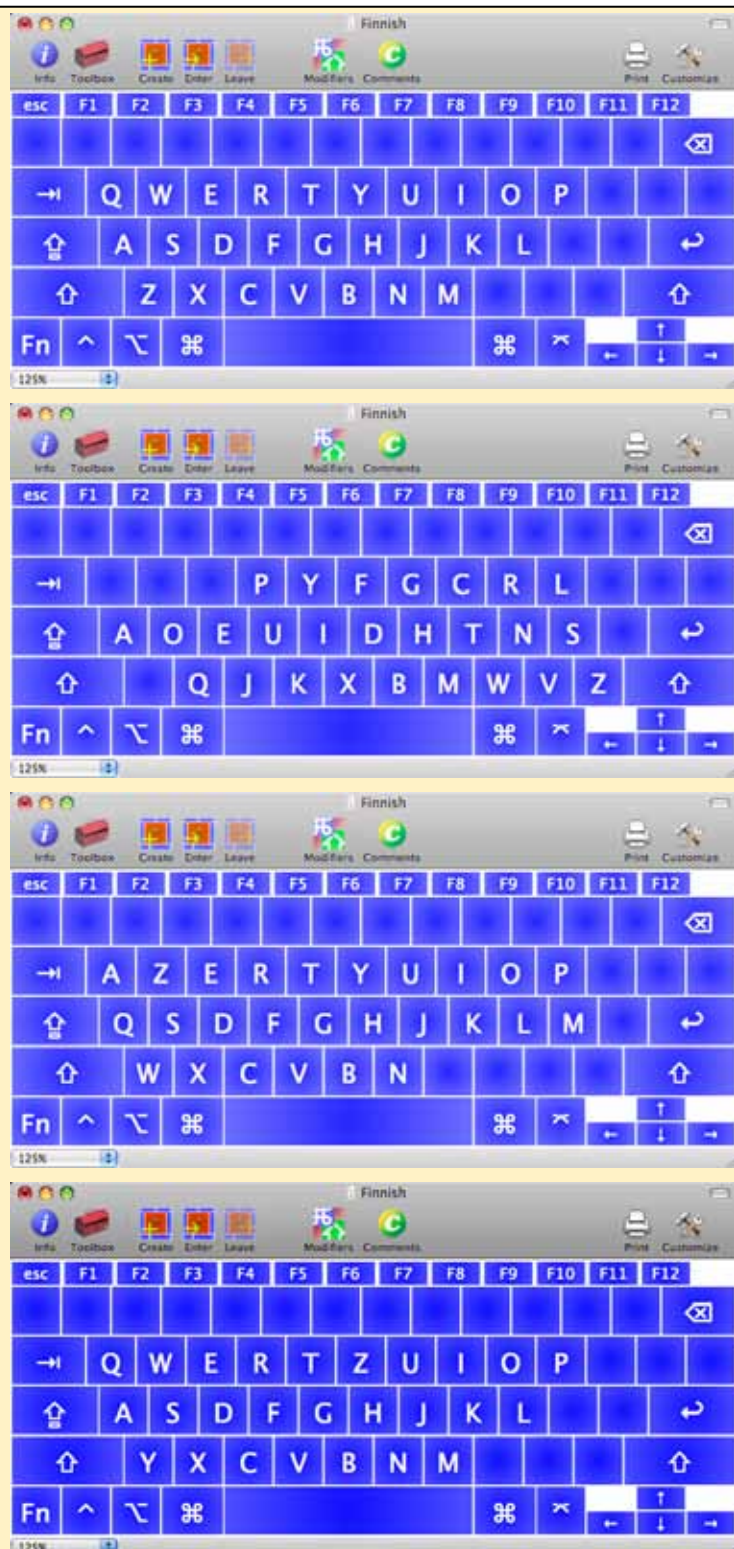
The empty key map is pretty straightforward: it provides nothing beyond the standard output for the “control keys”.

The QWERTY key maps provide the traditional English language keyboard layout, in upper and lower case.

The Dvorak key maps provide the Dvorak version of the English language keyboard layout, again in upper and lower case.

The AZERTY key maps provide the traditional French language keyboard layout, in upper and lower case.

The QWERTZ key maps provide the traditional German language keyboard layout, in upper and lower case.



combination. Once you click OK, the new row should appear in the window.

### **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 drawer.

## **5.8. Installing a keyboard layout**

When you have created a keyboard layout, you need to install it before it can be used.



Installation is quite simple, being a matter of putting the keyboard layout file into the correct folder. However, there are some options to take into consideration.

### 5.8.1. 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. 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. Press option and select Go from the Finder menu 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 be an administrator 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”.

### 5.8.2. Keyboard layout bundles

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 bundle.

Another issue is that support for “press and hold”, introduced with OS X 10.7 (Lion), requires a bundle 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 bundle.

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 localized names for any of the keyboard layouts in the bundle.

When you save a keyboard layout, you can save it as a stand-alone XML file, which is the default, and you would use ⌘-S to do so. You can use ⌘-S for Save As..., which allows you to save the keyboard layout to a new location. To save the keyboard layout as part of a bundle, you can use Save in Bundle... (⌘-S) or for Save As Bundle... (⌘-S). Save in Bundle... saves the keyboard layout into an existing bundle (it is up to you to choose a keyboard layout bundle), which Save As Bundle... saves the keyboard layout as a new bundle.

You can associate an icon file with your keyboard layout whether or not it is saved as a bundle. If it is not, then the icon file is saved in the same folder as the keyboard layout. Sim-

ply choose Keyboard > Attach Icon File..., which will bring up a dialog to open an icon file.

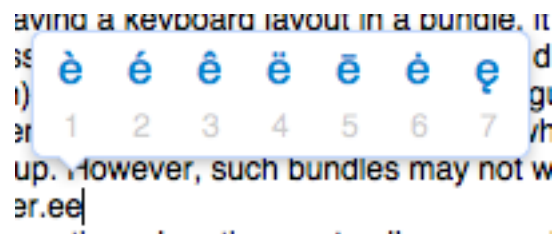
If your keyboard layout is to be saved in a bundle, you might want to associate some version information with it. Choosing Keyboard > Set Version Info... brings up a dialog which allows you to set three different version strings, Build Version, Bundle Version, and Source Version String. The first two are normally numbers, while the last can be almost anything. In practice, you can put whatever you want in these, as they are stored as strings.



A dialog box titled "Set Version Info" with three text input fields. The first field is labeled "Build Version:" and contains "1.0". The second field is labeled "Bundle Version:" and contains "1.0". The third field is labeled "Source Version String:" and contains "1000". At the bottom right are "Cancel" and "OK" buttons.

To create an icon file, you draw it in a graphics program, and then convert it to the icns format. This can be done with Icon Composer, which is installed with Apple's Developer Tools, or with one of several icon editors available on the internet.

With a bundle, you can choose to specify the "intended language" (in Apple's terminology). This allows you to tell the operating system which language to use when invoking the "press and hold" feature. For each language supported by Apple, there is a collection of variants which will be supplied with certain key presses. In OS X 10.7 (Lion), this is presented as a pop-up window with numbered alternatives.



Only certain languages are supported by Apple. The full list is given in chapter 10, 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 choose Keyboard > Set Keyboard Language... allows you to set values for all four, though only the language is required. At present, Apple supports only one case where specifying the script is useful, that of the Serbian language, which can use either Latin or Cyrillic script. No variants are currently supported, and, in fact, those listed by Apple are non-standard, so there is little reason to specify a variant.



A dialog box titled "Set Keyboard Language" with four sections: "Language (required)", "Script (optional)", "Region (optional)", and "Variant (optional)". Each section has a dropdown menu and a list of options. The "Language" section has "eng" selected. The "Script" section has "Script" selected. The "Region" section has "aus" selected. The "Variant" section has "Variant" selected. At the bottom right are "Cancel" and "OK" buttons.

Language (required)	Name	Code
eng	Bengali	bn
	English	en
	Antigua and Barbuda Creole English	aq
	Assam Heneq	akh

Script (optional)	Name	Code
Script	Alak	Alak
	Arabic	Arab
	Imperial Aramaic	Armi
	Armenian	Armn

Region (optional)	Name	Code
aus	Austria	AT
	Australia	AU
	Australia and New Zealand	053

Variant (optional)	Name	Code
Variant	Late Middle French (to 1600)	16_mic
	Early Modern French	16_cad
	Traditional German orthography	1901
	"Academic" ("governmental...ussian as codified in 1959	19_cad

In the dialog, you must choose a language, so you cannot set your keyboard to have no language if you save it in a bundle. If you do not set the language with this dialog, the language set for your operating system will be set.

## 6. 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.

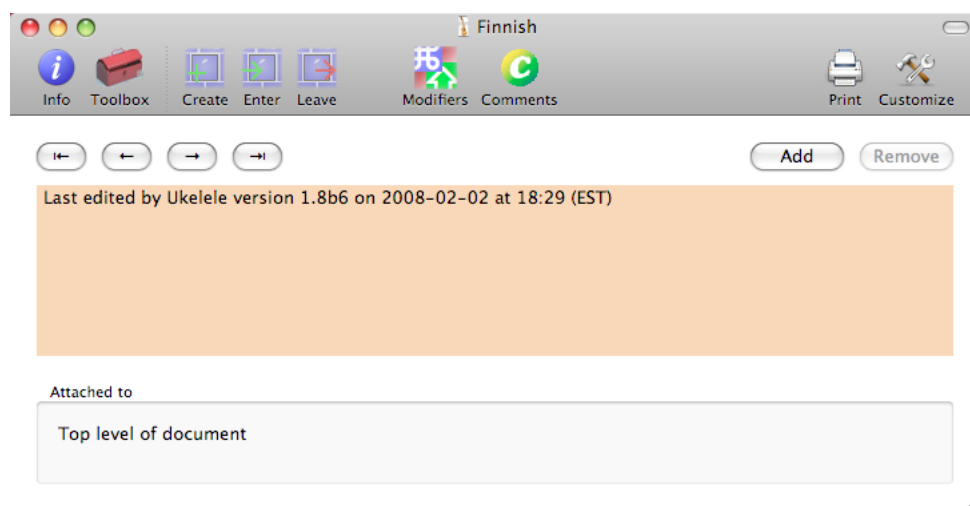
### 6.1. Comments

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. 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 with the comment editor. You open it by choosing "Show Comments" from the View menu, or clicking the "Comments" button on the toolbar. This changes the window to show the comments, 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 is visible, you choose "Add Comment..." from the Keyboard menu. You then type or click the key you wish to add the comment to (with whatever modifiers you need). The comment editor is then opened with a blank comment, which will show the key element in the bottom pane.



Click the Comments button again to return to the keyboard view.

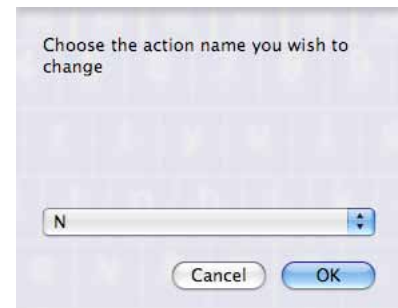
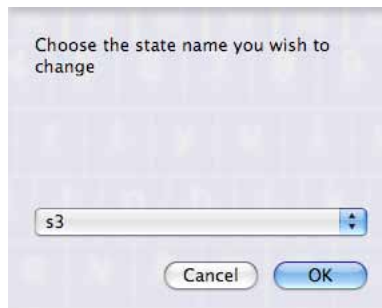
## 6.2. Changing dead key state and action names

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 8 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.

## 6.3. 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. You can't undo this action. 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, but it is still something that you cannot undo, and so be careful when doing it.

## 6.4. 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.

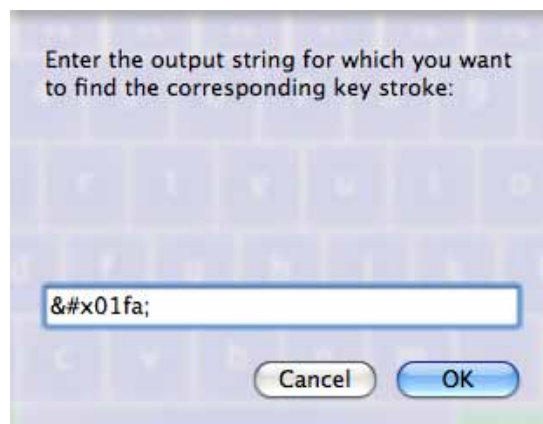
## 6.5. Searching for an output string

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 command (in the Edit menu) 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 `&#x01fa;` for Å (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 the status bar at the bottom of the window. 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, the status bar will tell you. 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.

## 7. 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.

### 7.1. Windows

Ukelele uses one window per keyboard layout, which has a drawer at the bottom for modifier combinations, and which can appear as either the keyboard or the comments. There are two floating windows: the info inspector and the toolbox.

#### 7.1.1. Keyboard layout window

This is the main window for interacting with Ukelele. 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.

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

#### What is the active keyboard?

All computers will have at least one keyboard attached (unless you have a headless server, which you probably wouldn't be using for running Ukelele, anyway). It's possible to have more than one keyboard attached, since almost all keyboards these days are attached by USB or wirelessly via Bluetooth. At any given time, there is one keyboard considered as active. In general, that is the one that was last used for keyboard input.

One thing that can be disconcerting is that the active keyboard can be something other than the actual keyboard attached to the computer. This occurs in a single circumstance on some versions of Mac OS X: when you start an application, no keyboard is considered to be active until the keyboard gets used. So, if you start Ukelele with the mouse, and then open or create a new keyboard layout without using the keyboard, the operating system tells Ukelele that there is no active keyboard. In this case, Ukelele will choose a default keyboard. Unless specified otherwise by the preferences, this will be the original USB ANSI keyboard (with keypad). It is also possible to specify that a particular keyboard is always used, regardless of what keyboard is attached to the computer.

Another issue is where there is a third party keyboard attached to the computer. In general, there will be no KCAP resource for such keyboards, and so Ukelele will choose a default keyboard type to use for display.



as you press or release modifier keys. If you hold a key down, that will also be shown in the window.

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) will be shown as its hex representation, such as `&#x009c;` for the “string terminator” 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.

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.

The exact appearance of the keyboard window depends on which colour theme is active. There are always at least two colour themes available in Ukelele, named Default and Print. The screenshots in this manual show the Default theme, and further discussion will generally refer to this theme. Themes can be chosen from the View menu, for which see below. You can also make a theme the default in the Preferences window.

A key shown with the dead key colour scheme (red with the Default colour theme) 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) from a source such as the Character Palette 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.

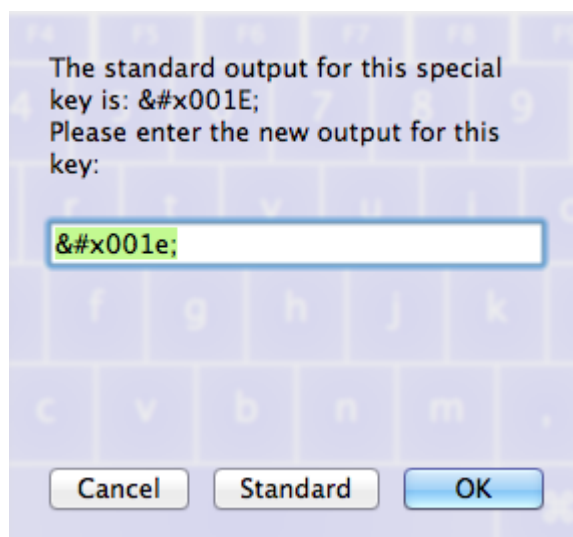
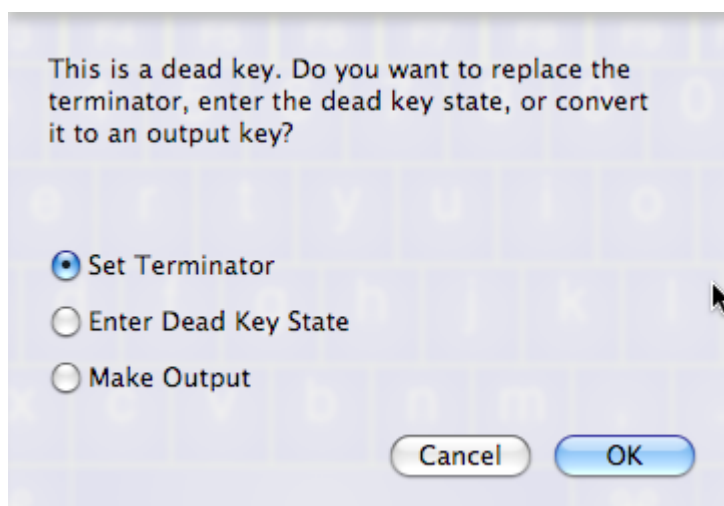
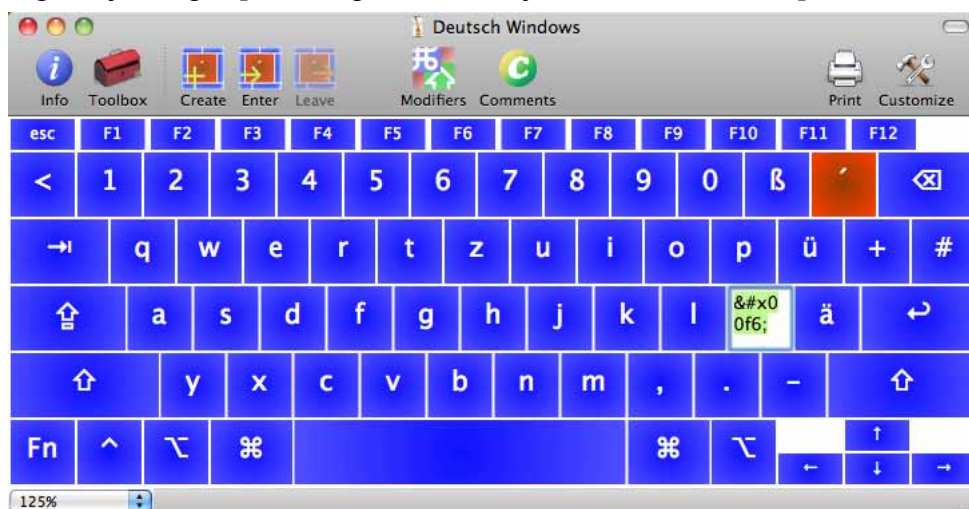
If you drag text onto a dead key, you are offered two options: you can make the dragged text the new output of the key, thus changing it into an ordinary key rather than a dead key, or changing the terminator of the dead key state.

Double-clicking a key brings up a dialog that allows you to enter new output for the key. This can be a sheet dialog or within the window (as shown). 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 `&#x<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.

If you double-click a dead key, you are offered three choices: changing the terminator of the dead key state, entering the dead key state, or changing the dead key into an ordinary key and assigning new output.

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 dou-





ble-click these keys to change their output. When you do, and the preferences are set for a sheet to appear, the dialog 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.

In a couple of cases, you may be asked to choose two keys, such as when swapping two keys. When the first has been chosen, the key will be shown with the “selected” colour (yellow in the Default colour theme) to indicate that it has been chosen.

In the lower left corner of the window is a pop-up menu 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 50% and 800%.

### 7.1.2. Modifier combinations drawer

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.

Modifier combinations are examined and edited from the modifier combinations drawer, which is shown or hidden by selecting “Show Modifiers” or “Hide Modifiers” from the View menu, or pressing  $\hat{\text{C}} \mathbb{M}$  (command-shift-M), or clicking the “Modifiers” button on the toolbar.

Each row in the table presented in the drawer 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. To save space, these are listed slightly differently:

Values	Listing
Both up	Both Up
Both down	Both Down
Both either up or down	Either Down or Up
Either left or right down	Either Down
Left up and right down	Left Up, Right Down
Left down and right up	Left Down, Right Up
Left up and right either down or up	Right Down or Up
Left down and right either down or up	Left Down, Right Down or Up
Left either up or down and right up	Left Down or Up
Left either up or down and right down	Left Down or Up, Right Down

Note also the first column, labelled “Set”. It is possible to have more than one modifier combination use the same set of outputs in the keyboard layout. This is accomplished by having two or more rows in the table with the same set number. All rows with the same set number map to the same set of outputs. The collection of rows with the same set number is called a modifier map, and the set of outputs associated with it is called a key map.

At the bottom of the drawer 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

Set	Shift	Caps Lock	Option	Command	Control
0	Both Up	Up	Both Up	Down...r Up	Both Up
0	Either Down	Down...r Up	Both Up	Down	Both Up
1	Either Down	Down...r Up	Both Up	Up	Both Up
2	Both Up	Down	Both Up	Down...r Up	Both Up
3	Both Up	Down...r Up	Either Down	Up	Both Up
3	Both Up	Down	Left Down o..., Right Down	Down	Both Up
4	Either Down	Down...r Up	Either Down	Up	Both Up
4	Either Down	Up	Left Down o..., Right Down	Down	Both Up
4	Left Down o..., Right Down	Up	Either Down	Down	Both Up
4	Either Down	Down	Either Down	Down...r Up	Both Up
5	Both Up	Up	Either Down	Down	Both Up
6	Either Down or Up	Down...r Up	Either Down or Up	Up	Either Down
6	Either Down or Up	Up	Either Down or Up	Down...r Up	Either Down
6	Either Down	Down	Either Down	Down	Left Up, Right Down
6	Either Down	Down	Right Down or Up	Down	Either Down
6	Right Down or Up	Down	Either Down	Down	Either Down

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 edit a modifier combination, double-click the corresponding row. This brings up the modifier combinations dialog. Select the combination that you want and click OK, and the row will show the new value.

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

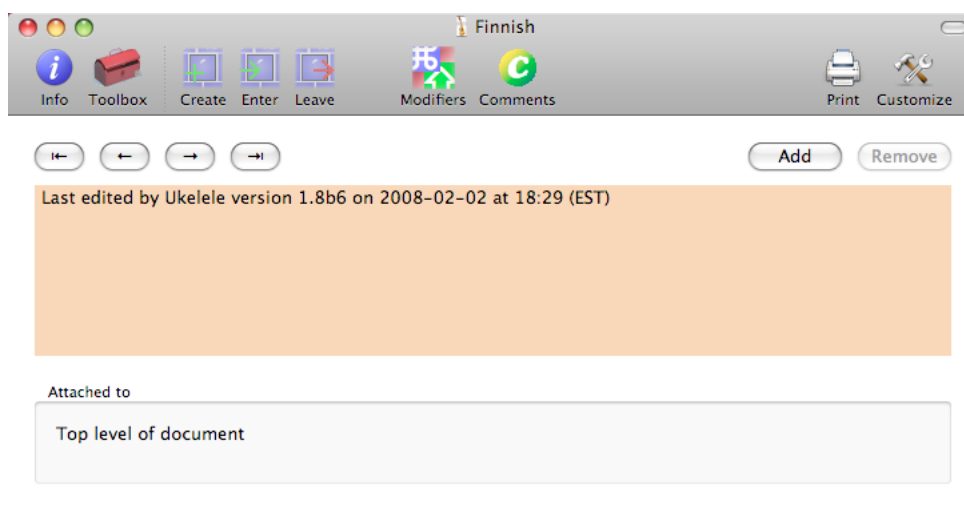
To add a modifier combination, you can either add one to an existing set, or create a new set. These options are chosen from the pop-up menu that appears when you click the “+” button. You can only add a set to an existing set if you have already chosen a row in that set.

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, 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 standard French layout (AZERTY), or the standard German layout (QWERTZ).

### 7.1.3. Comments editor

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.

In the comments editor, which is opened by choosing “Show Comments” from the View menu or clicking the “Comments” button on the toolbar, 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, delete and edit 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 KCHR 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.

#### 7.1.4. The info inspector

The info inspector is shown by either clicking the Info button on the toolbar, choosing View > Show Info, or pressing  $\mathbb{H}$ -I. It is a floating window with three sections, two of which can be collapsed by clicking on the disclosure triangle.

The first section shows the output of the key that is currently under the pointer. This is given as a Unicode code point in the U+xxxx format, plus a description drawn from the Unicode database. If you have more than one character output, some of it may be clipped if the descriptions happen to be long.

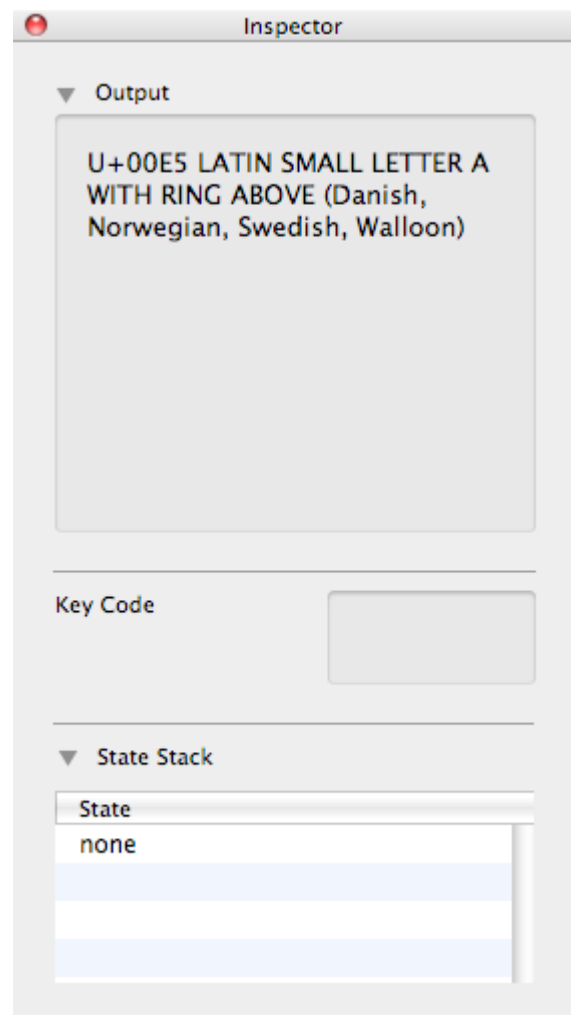
If the key under the pointer is a dead key, then the output section lists the name of the dead key state and the terminator.

The second section shows the key code of the key that is currently pressed. 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.

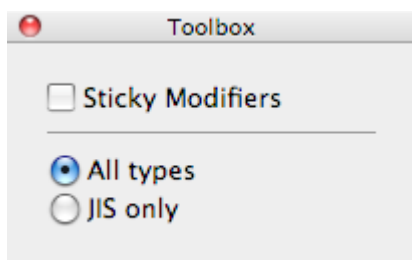
The last section is the “state stack”. This shows the stack of dead key states. At the bottom of the stack is the special state “none”, which is used to indicate that there is no dead key state active. As you enter a dead key state, its name is pushed onto the top of the stack, and popped off when you 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.

#### 7.1.5. 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 pair of radio buttons 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 toolbox has two radio buttons affecting 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 radio button: JIS only means that you are editing the JIS overrides, whereas All types edits the underlying output for all keyboard types.

## 7.2. Menus

The menus in Ukelele are mostly standard menus. The main menus 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.

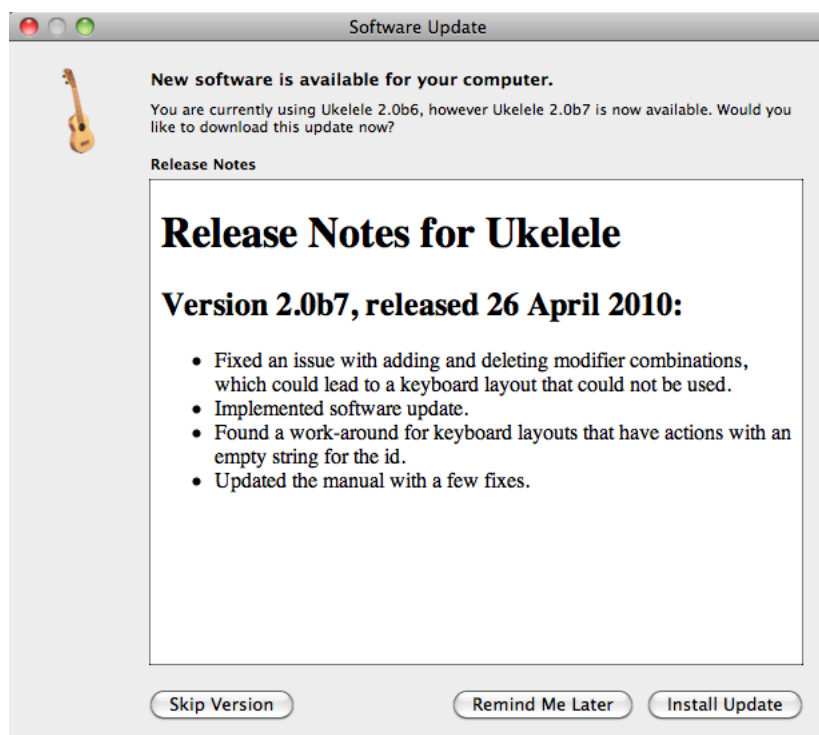
### 7.2.1. Ukelele menu

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

Software Update is available from Ukelele 2.0b7. 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 in your web browser. You will then have to install it yourself.

The preferences dialog has several panels: General, Fonts, Window and Dead Keys.

The General preferences panel allows you to



choose what action Ukelele should take when launched. The default is to create a new, empty keyboard layout. The other options are to bring up a standard open file dialog, or to do nothing.

Below the startup options there is a checkbox which offers an option only useful in Mac OS X 10.7 (Lion) or later. By default in Lion, applications reopen the documents that were open when the application closed. This can be turned off with this option. If this is turned on, and you had open windows when you last closed Ukelele, then they will be reopened, and the startup action will not happen.

At the bottom of the General preferences panel is the option for using a single click to edit keys. By default, this is off, and you have to double-click a key in the window to bring up the dialog to edit the key's output. When this is set to on, then a single click on a key opens the dialog.

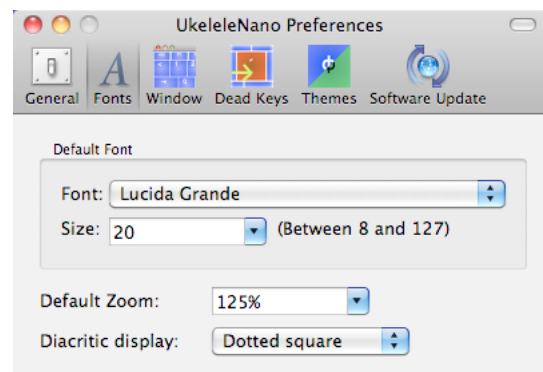
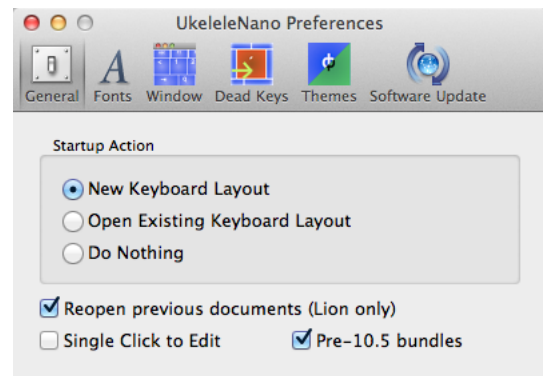
The last option on this pane is for keyboard layout bundles, with the option of saving the bundle so that they can be used on pre-10.5 systems, that is, Mac OS X 10.4 (Tiger) or earlier. This only changes the bundle ID, which is used by the operating system to identify what kind of bundle it is. In future 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.

The Fonts panel allows you to specify the default font and size that you want for the keyboard display. The default is Lucida Grande 20 point, which is a fairly good choice in most cases, as Lucida Grande contains more Unicode characters than most other fonts. The size of 20 point is suitable for 100% zoom. If you change the default zoom, you may want to change the default font size as well.

The default zoom is used for new windows. You may choose a value from the list (100%, 125%, 150%, 200%, 250%, 300%, 350%, 400% or Fit Width) or enter a custom percentage value, which must be between 50% and 500%. If you enter a value outside that range, it will be restricted to the range. The Fit Width value will make the window as wide as will fit on the screen, with the whole keyboard visible.

The last option in the Fonts panel is the character to be used when displaying combining diacritics. Unicode has a range of characters which combine with the preceding character to form a complex character. To make these easier to see on the keyboard layout, they are shown as combined with another character. You can choose which one you want from the popup menu. Note that this only affects the display, not the actual output of the key.

The Windows panel offers you the choice of what kind of keyboard Ukelele should use for display when the hardware keyboard is not recognised by the system. This offers you the same choices as choosing Keyboard Type... from the View menu, so that you can choose both the keyboard type and the coding type as appropriate.



✓ Dotted square  
Dotted circle  
White square  
White rectangle  
Space

At the bottom of the panel are two checkboxes. The first allows you to make Ukelele always use the keyboard type that you select. If this is checked, then Ukelele will not check what the hardware keyboard is, but will display a keyboard window using the type specified here.

The second checkbox allows you to choose whether double-clicking a key in the keyboard window will bring up a sheet for entering the new output, or edit it in the same window.

The Dead Keys panel allows you to choose how Ukelele handles creating a dead key. If the first checkbox is checked, then a dialog will appear to ask for a terminator when creating a new dead key. Otherwise, the new dead key will be created with the empty string as the terminator.

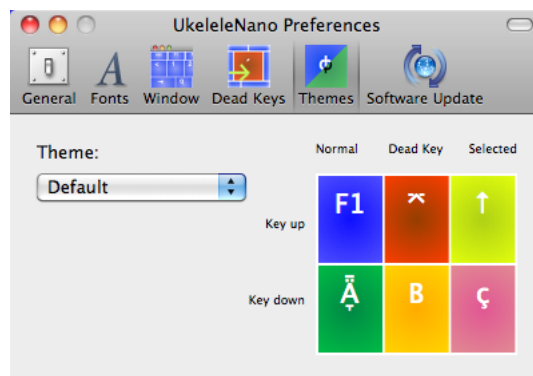
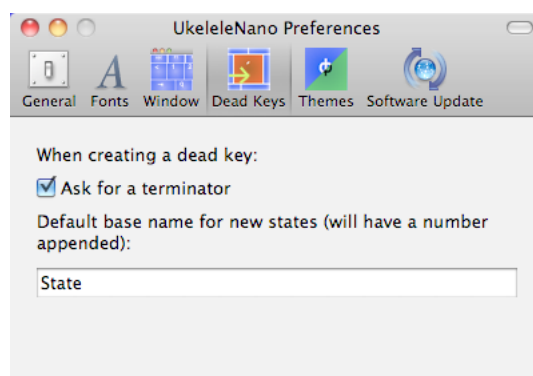
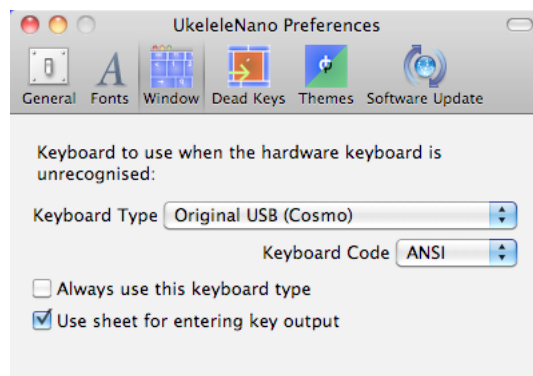
The base name for a dead key state is used for generating automatic names for dead key states. The base name has a number appended to it to make it unique. It is this name that is filled in for you when you create a dead key, and you can either change it in that dialog, or you can change the name of the dead key state later by choosing Change State Name... from the Keyboard menu.

The Themes panel allows you to choose the colour theme that applies to the window when it is created. This includes the colour scheme for the keys in their various states, the background colour for the window (the area where there are no keys), and the colour of the text on the key for each possible state.

Each key has six possible states. It may be up or down, and it may be a normal key, a dead key, or the selected key (for some operation such as swapping keys). For each of the states, you can specify two colours, inner and outer, a gradient style, and the text colour. With the radial gradient style, the key will be drawn with colours varying from the outer colour at the corners to the inner colour at the centre of the key. A linear gradient will vary from the outer colour at the top of the key to the inner colour at the bottom of the key. If you choose no gradient, then the outer colour will be a frame, and the inner colour the solid colour for the rest of the key. If you choose both inner and outer colours to be the same, then the key will appear as a solid colour with all three gradient types.

When you click on the different colour pickers, a standard colour palette will appear, allowing you to specify the colours with any of the methods available in the colour palette. The colour picker that is active will be shown with a highlight.

You can customise the current colour theme, and then save it with the popup menu. If you do not save the customisations, then the modified colour theme will not be saved in the preferences.



The Software Update panel allows you to choose how often Ukelele will check for updates. The default is to check once a week. You can also check immediately, or disable checking. Immediate checking can also be done from the Ukelele application menu.

### 7.2.2. 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  $\text{⌘N}$ ), Ukelele will create a new, empty keyboard layout. This has only special key output, and a basic set of modifier combinations.

#### New Based On...

Choosing New Based On... (or pressing  $\text{⇧⌘N}$ ) brings up a standard open file dialog, which lets you choose an existing keyboard layout. When you choose the keyboard layout, it is opened as a copy, so that it is not saved, but has the same name as the original keyboard layout.

#### 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 (KCHR and uchr) keyboard layouts into a form that can be modified with Ukelele. The resource-based keyboard layouts may not be usable on the latest versions of Mac OS X, which could make this difficult.

Please note that this option runs some Apple tools. The one that converts KCHR resources is Power PC only code, and so requires that Rosetta be installed if you are running Ukelele on an Intel Mac, and so is not usable on Mac OS X 10.7 (Lion) or later.

#### Open...

This brings up a standard open file dialog. You should only open keyboard layout files, that is, those with the extension .keylayout, or bundles which contain keyboard layouts. It is possible to open other bundle types, but should get an error message when you do.

#### Open Recent

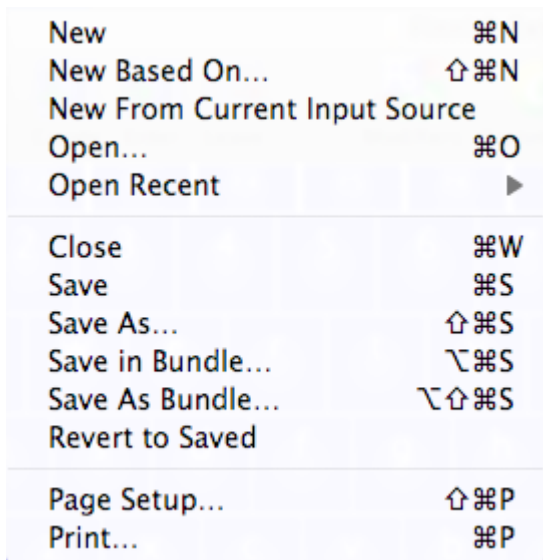
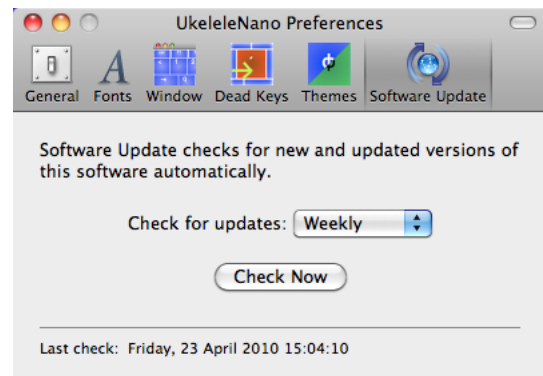
This submenu lists the ten most recent keyboard layouts that have been opened.

#### Close

As usual, this command closes the current keyboard layout. If the keyboard layout has been changed but not 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.



### Save As...

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  $\text{⌘}S$ .

### Save in Bundle...

You can save a keyboard layout into a bundle that already contains one or more keyboard layouts. If you have associated an icon file with it (like a flag icon), then the icon is also saved in the bundle. The language for the keyboard layout will also be saved in the bundle. The keyboard shortcut is  $\text{⌘}S$ .

### Save As Bundle...

To create a new bundle containing this keyboard layout, choose this command, or type  $\text{⌘}S$ . This will bring up a standard save dialog, and you can save the keyboard layout in a new bundle that you specify. The default name for the bundle is the same as that of the file name for the keyboard layout. If you have associated an icon file with the keyboard layout, it is saved in the bundle as well. The language for the keyboard layout is automatically saved in a bundle.

### Revert to Saved

This command discards all changes to the frontmost document since it was last saved. It is the same as closing the document without saving the changes, and then opening the file again. It is only enabled when the current document has unsaved changes. A warning dialog is shown when you choose this, to avoid unintended loss of data.

### Page Setup...

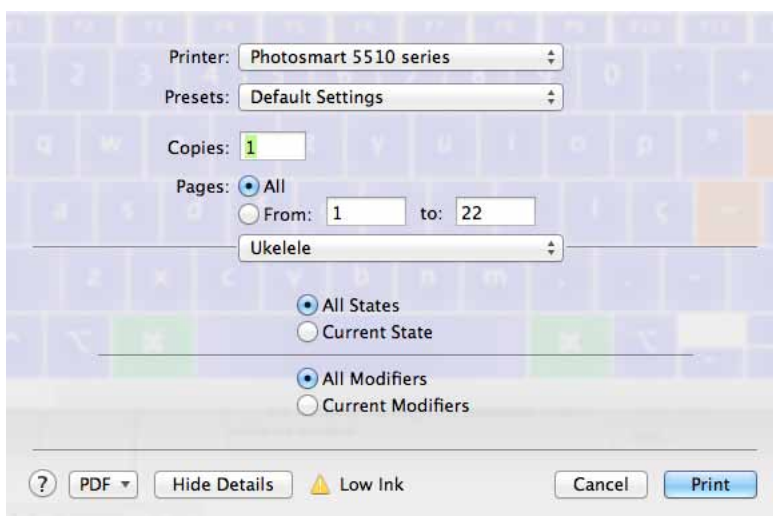
This is the standard Page Setup dialog.

### Print...

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, not whatever colour theme is currently active for the keyboard window.

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. Note that the page numbers are not automatically updated when you change these options.

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.





### 7.2.3. 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  $\text{⌘}Z$ .

#### Cut

This is the standard command, and is only enabled when a text field in a dialog is active, and text is selected.

#### Copy

This is the standard command, and is only enabled when a text field in a dialog is active, and text is selected.

#### Paste

This is the standard command, and is only enabled when a text field in a dialog is active, and there is text on the clipboard.

#### Delete

This is the standard command, and is only enabled when a text field in a dialog is active, and text is selected.

#### Select All

This is the standard command, and is only enabled when a text field in a dialog is active, there is text in the text field and not all the 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,  $\text{⌘}X$ ) you get the dialog shown at right.

You can choose the key that you want to cut either by specifying its key code (in which case you choose "Enter key code" and enter the code in the text field) or by choosing "Press or click key" and then either pressing the key or clicking the key in the Ukelele keyboard window.

Undo Change Output	$\text{⌘}Z$
Can't Redo	$\text{⌘}Z$
Cut	$\text{⌘}X$
Copy	$\text{⌘}C$
Paste	$\text{⌘}V$
Delete	
Select All	$\text{⌘}A$
Cut Key...	$\text{⌘}X$
Copy Key...	$\text{⌘}C$
Paste Key...	$\text{⌘}V$
Find...	$\text{⌘}F$



### Copy Key...

This command functions much as Cut Key..., though without clearing the key from the keyboard. It has the keyboard shortcut  $\backslash \text{⌘} C$ .

### 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, you are presented with a dialog asking how you want to choose the key, either directly or by key code. The keyboard shortcut is  $\backslash \text{⌘} V$ .

### Find...

The Find command allows you to search for keystrokes that will produce a given string. When you choose the command (or use the keyboard shortcut,  $\text{⌘} F$ ), 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  $\&\#x03b5$ ; 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 status bar at the bottom of the window. 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.

### Special Characters...

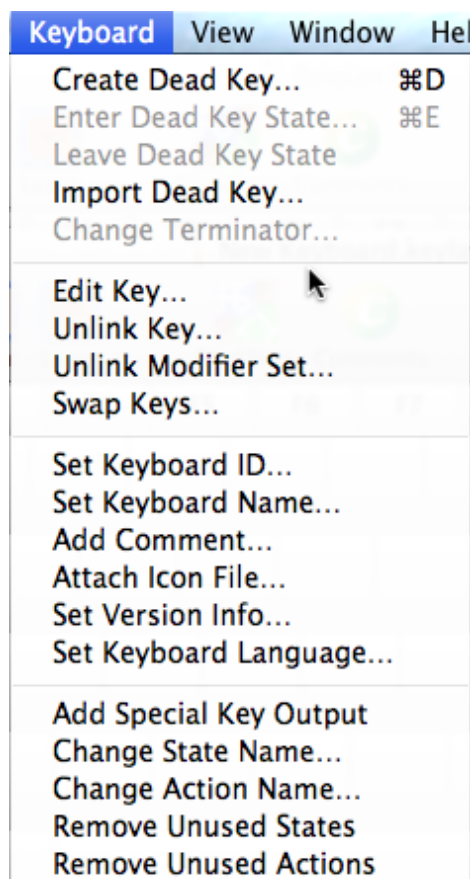
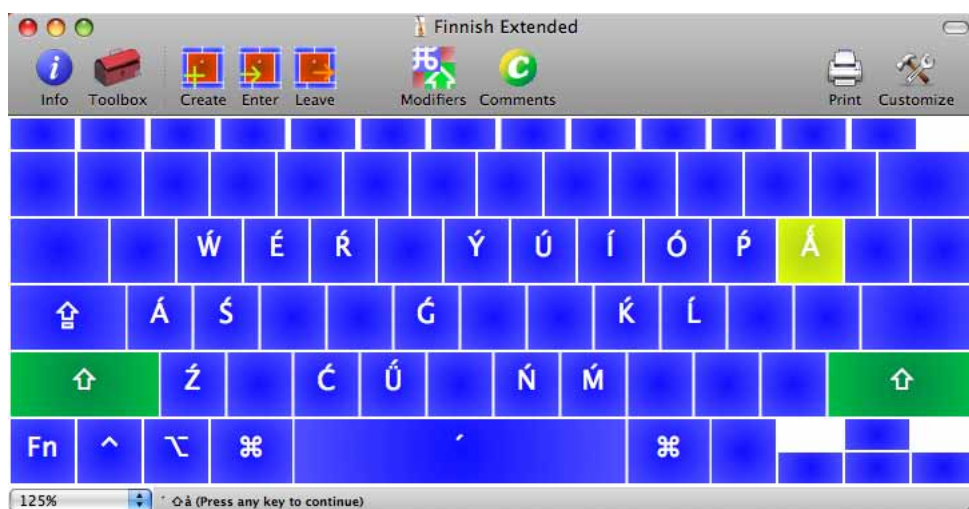
This menu item shows the character viewer. It is equivalent to choosing Show Character Viewer from the input menu, and is available even if the character palette has not been enabled in the Language & Text (International on Leopard or earlier) preference pane.

#### 7.2.4. Keyboard menu

The Keyboard menu has most of the commands used for editing a keyboard layout.

### Create Dead Key...

The whole area of dead keys is complex, and is



dealt with more in section 5.6. Choosing Create Dead Key... starts the process of creating a new dead key, and waits for you to tell it what the dead key should be. In the status bar at the bottom of the window, you will see the instruction to press or click the new dead key.

When you press or click the new dead key, the modifiers that are active at the moment are the relevant ones. If Sticky Modifiers is on, then it is those modifiers that are shown as pressed in the keyboard window. Otherwise, it is those modifiers that are pressed when you either press the key, or click the key in the keyboard window.

After you press or click the new dead key, you will see a dialog appear. You have two choices at this point. One is to create a new dead key state, which is done by clicking the first radio button. The other is to use an existing dead key state. To do this, click the second radio button and choose the name of the dead key state from the pop-up menu.

If you choose the second option, then choosing the existing state completes the process, and the dead key has been created.

If you choose the first option, then you need to set a name for the new dead key state. Ukelele suggests a name, based on the base name set in preferences, but you are free to use another name. If you choose a name that already exists, Ukelele tells you that, and cancels the process of creating a dead key.

What happens after that depends on how your preferences are set. The default is that you will be asked to provide a terminator for the new dead key state. In the preferences, you can set it so that Ukelele will not ask, but set the terminator to the empty string.

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 main Ukelele 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. If there is only one dead key state defined, then the command is disabled once you have entered that dead key state.

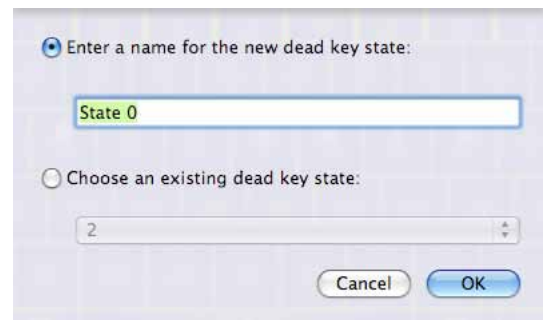
### **Leave Dead Key State**

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 Ukelele window will update to show outputs in the previous dead key state.

### **Import Dead Key...**

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 lay-





out 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.

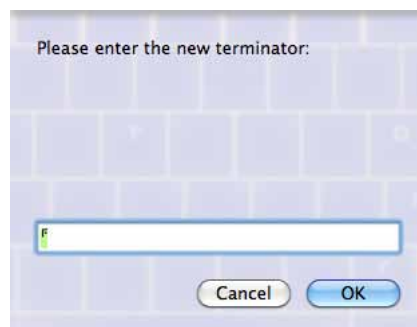
When you choose this command, Ukelele asks you to open the other keyboard layout by showing you a standard open file dialog. Ukelele then checks that the modifier key combinations are the same, and then asks you to identify the dead key state you want to import. It asks for the name of the dead key state 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 suggesting the same name as in the source keyboard layout.

Note that this command cannot be undone by using the Undo command. However, you can change the dead key you just created by making it into an ordinary key (by assigning output to it), and you can delete the imported dead key state by choosing Remove Unused States from the Keyboard menu.

### **Change Terminator...**

This command is also only available when there is a dead key state active. It allows you to change the terminator for that 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.

A dialog appears, allowing you to specify a new terminator. In the text field is the current terminator and you can supply the new terminator by any method: typing it, entering the Unicode code point, pasting, entering it via the character viewer, drag and drop, etc.



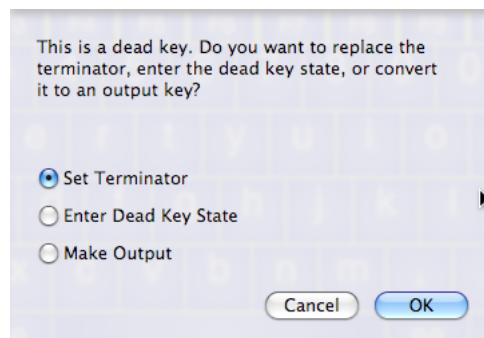
put  
key  
tor,  
od:  
it,

### **Edit Key...**

When you choose this menu item, you get a dialog asking you how to select which key you want to edit. The alternatives are the same as in several other places: either pressing or clicking the key, or giving the key code. If you choose press or click, then the status bar will show you that you need to press or click a key. If you give a key code, you go directly to the dialog. Either way, you get the dialog to edit the output of a key.

If you select a dead key, you are presented with a dialog offering you the choice of setting the terminator for the dead key state, entering the dead key state, or converting the dead key to an output key.

Selecting this menu item and choosing the key is equivalent to double-clicking the key in the keyboard window.



### **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. 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.

Choosing this menu item brings up a dialog which tells you to type or click the key that you want to unlink. After you click OK and dismiss that dialog, the next key you type is the relevant key. In the normal state, the modifier keys that are pressed when the key is typed are those that are in effect. So, typing  $\text{⌘}$ -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.

Apart from typing the key, you can click the key in the Ukelele window. The same rules apply as to what modifier keys are in effect.

### 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.

Because it's hard to type a modifier set, the way to tell Ukelele what modifier combination you want is to type any key with the set of modifier keys you want. So, if you want to unlink the caps lock set, press caps lock and type any key. Again, you can do this with point and click on the keyboard in the Ukelele window.

### 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 either by referring to the keys directly (by pressing the key or clicking the key in the Ukelele window) or by their key codes. A dialog comes up for each of the keys when you choose this command, and you click the appropriate button for how you would like to select the keys.

Once you choose the first key, and it will be highlighted with the "selected" colour from the current colour theme. You then choose the second key, and the swap happens and the highlight disappears.



### Set Keyboard ID...

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 renum-

ber keyboard layouts that have the same ID. However, some anecdotal evidence exists to imply that keyboard layouts work better when you give them distinct IDs.

However, 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.

Starting from version 2.2, when opening a keyboard layout, Ukelele checks that the keyboard ID is in the right range, and fixes it if necessary.

For non-Unicode keyboard layouts, you should assign IDs within the appropriate range for the script, from the following table:

Script	ID range
Roman	2–16383
Japanese	16384–16895
Simplified Chinese	28672–29183
Traditional Chinese	16896–17407
Korean	17408–17919
Cyrillic	19456–19967
Central European	30720–31231

### Set Keyboard Name...

The name of a keyboard layout is what you see in the Input menu (the menu with the flag), and in the Input tab of the Language & Text (International on Leopard or earlier) preference pane. A new keyboard layout has a name “New Keyboard”, which can be set using this menu item.

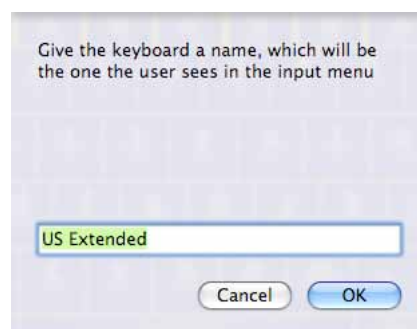
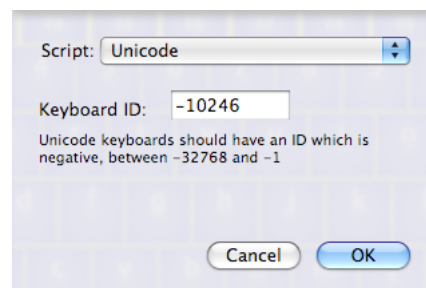
If you don't set a name with this command, then Ukelele will assign a name based on the file name when you save it.

### 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. When you choose this menu item, Ukelele will ask you to specify the key by pressing or clicking. After you do so, then the comments pane is shown, with the new comment present, empty and ready for you to edit.

### Attach Icon File...

When you have installed a keyboard layout, you will see that it has an icon in the Language & Text (International on Leopard or earlier) preference pane and in the Input menu. If you have just the keyboard layout file installed, then a generic keyboard icon is shown. If



you want to have a more distinctive icon, such as the flag icons that the system keyboard layouts have, then you need to attach an icon to your keyboard layout. This command accomplishes it.

Once you choose this command, Ukelele brings up a standard open file dialog for you to choose the icon file. It has to be an icon file, in Apple's "icns" format. There are several applications available for generating these files, including Icon Composer, part of Apple's developer utilities.

The effect of associating an icon file is not seen until you save the keyboard layout. If you save it as a keyboard layout file, the icon file is copied to the same folder, with the same name as the keyboard layout file, but with the extension ".icns" rather than ".keylayout". When installing the keyboard layout, both files should be copied to the Keyboard Layouts folder. If you save the keyboard layout into a bundle, then the icon file is saved in the appropriate place within the bundle. Either way means that the correct icon should appear in Language & Text (International on Leopard or earlier) preferences, and in the Input menu.

### Set Version Info...

When you create a bundle, then there is a selection of version information saved with it. The exact format of this information is really up to you, but Ukelele allows you to specify three specific items: the build version, the bundle version, and the source version string. These can really be of any kind, but generally you would stick to standard version number conventions, with numbers like 1.0.2 or 1.3b2 for the build and bundle versions. The source version is really up to how you want to specify it, as it is more of a string than a number.

Ukelele displays a dialog allowing you to save this information. It is only saved if you save the keyboard layout in a bundle, and it refers to the bundle, not the keyboard layout within the bundle.

### Set Keyboard Language...

When a keyboard layout is saved in a bundle, a language is associated with it. By default, the language is that currently set in the operating system, usually the one you set when installing OS X. You can choose a different language by selecting this menu item.

It brings up a dialog in which you can set the language, script, region and variant. Only the language is required, and, with current versions of OS X, little is to be gained by setting the script (apart from Serbian) and variant.

In each section of the dialog, there is a search field which can be used to find entries quickly. The search is not case-sensitive, but is sensitive to diacritics, so to find Volapük, typing "volapuk" will not work.

For a list of languages and regions

that are supported, see chapter 10.

### **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 use the Enter Dead Key State... command, and pick a state and see if it's the right one.

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. You can't undo this, so be sure that it's what you want to do.

### **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. You cannot undo removing unused actions, so be sure that it is what you want to do.

### 7.2.5. View Menu

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

#### Show/Hide Toolbar

This shows or hides the toolbar in the keyboard window. The keyboard shortcut is `⌘T`. It is equivalent to clicking the lozenge button on the right hand end of the top of the window on OS X 10.6 (Snow Leopard) or earlier.



#### 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 Fonts

When you choose Show Fonts, the standard Mac OS X font panel appears. You can use this to change the display in the keyboard window. When you choose a font and/or a size, the window will update to use that font and/or size to display key output. If the output of a key doesn't fit, it will try a smaller size, 60% of the designated size. If that doesn't fit, then the key will look blank.

Note also that, since not all fonts contain all Unicode characters, the system will substitute another font that does have a given character, or display an error glyph if no active font contains that character.

Only font and size can be set with the font panel. Underline, strikethrough, font and document colour, and text shadow have no effect.

There can be a problem with the font panel if your primary system language is not English. The default collection is defined by the system language, and may reduce the fonts available to you. To see all fonts, you need to enable the All Fonts collection. If you only see the Family and Typeface columns in the font panel, you need to make it bigger so that the Collection column appears on the left, and you can choose All Fonts from there.

#### Show/Hide Info

The info inspector is a floating window that shows three types of information. The top section shows the output for the key under the mouse. It shows it as a Unicode code point in the U+xxxx format, followed by information from the Unicode Character Database. This contains names and other information about the character. If the key is a dead key, it shows the name of the dead key state and the terminator for that dead key state.

The second section shows the key code of the key that is currently pressed. If multiple keys are pressed, it shows the last key pressed. OS X limits the number of simultaneous keys that are recognised to six, so holding down more than six keys at once will only show the sixth.

The last section shows the "state stack". This is the stack of dead key states. The current state is at the top, and the last entry will always be "none", the special state name for

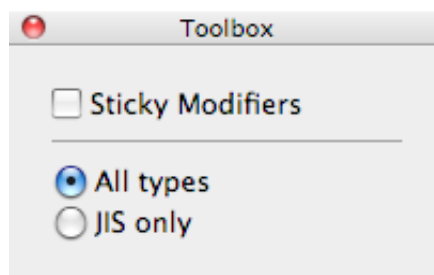
not dead key state being active.

You can collapse the first and third sections by clicking on the disclosure triangle. Note that in versions prior to OS X 10.7 (Lion) there is a cosmetic display bug with this, in that the display gets darker after the disclosure triangle is toggled closed and open.

### **Show/Hide Toolbox**

Ukelele has a small floating window with controls for two different situations, which is shown or hidden by this command, or by clicking on the Toolbox button on the toolbar.

The first control is a checkbox for setting “Sticky Modifiers” on or off. This is equivalent to choosing the Sticky Modifiers command from the View menu, for which you should see the section on this below.



The other control is a pair of radio buttons that are of use when you have a JIS keyboard as the displayed keyboard in the keyboard window. These determine how changes of output are applied. In the normal case, All types, output is changed for all keyboard types, that is ANSI and ISO as well as JIS. When JIS only is selected, the output is only changed for JIS keyboards. In other words, output you set becomes a “JIS override”.

The idea of a JIS override is that there are some conventions for Japanese keyboard layouts that assign output to certain keys that is different to US conventions. Apple’s system accomplishes this by allowing for JIS keyboards to have some behaviour override the standard ANSI/ISO keyboards. Normally, this is just a few of the keys on the keyboard, typically less than twenty. Other keys fall back to the ANSI/ISO case.

### **Show/Hide Modifiers Drawer**

This command opens or closes the modifier key combinations drawer for the current keyboard window, and is equivalent to clicking the Modifiers button on the toolbar. This allows you to see and modify modifier combinations for the keyboard layout. The operations available in the modifiers drawer are explained in section 7.1.2.

### **Show/Hide Comments**

The Show Comments command is equivalent to clicking the Comments button on the toolbar. It shows or hides the comments pane, allowing you to edit or inspect the comments associated with the current keyboard layout. The operations available are explained in section 7.1.3.

### **Colour Theme submenu**

The Colour Theme submenu allows you to change the colour theme displayed with the current keyboard window. You can choose from one of the existing ones, or you can customise one to the way you want it.

When you customise a colour theme, you are presented with a dialog that allows you to specify all the colours and the gradient styles used in the window. Each key can have six possible states, up or down, and either normal, dead key or selected (as part of a swap key operation). For each of these six states, you can choose two key colours, the text colour and a gradient style. The gradient style you choose will define the meaning of those two key colours.

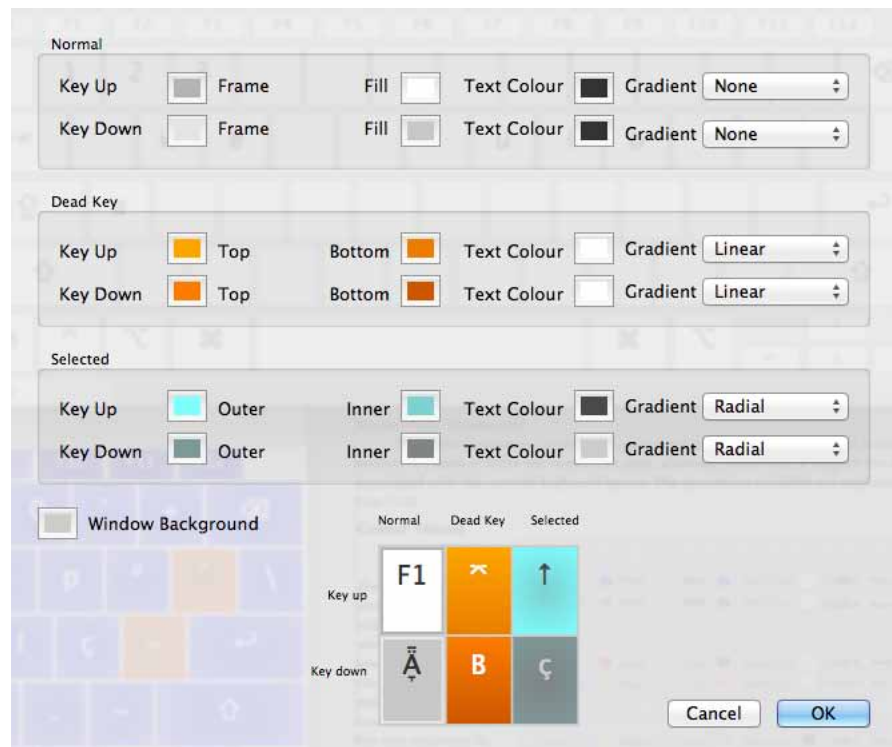


A radial gradient varies the colours from the inner colour at the centre of the key to the outer colour at the corners of the key. A linear gradient varies the colours from the



outer colour at the top of the key to the inner colour at the bottom of the key. When “None” is chosen as the gradient style, the key is drawn as a solid colour with the inner colour, and a two pixel frame with the outer colour. Note that, if both inner and colours are the same, then all three gradient types will appear as solid colours.

Apart from the keys, you can also change the window background colour, which is the colour shown in the spaces around the keys.



When you have customised a colour theme, a new entry will appear in the submenu, with the name of the original colour theme plus “(modified)”. The “Save Theme As...” menu item will also be enabled, allowing you to save the theme that you have now created. If you choose a name that is already in use, you have the option of replacing it with the new theme.

Two special colour themes are Default and Print. There is a version of each defined by Ukelele, but you can override them by customising a colour theme and saving with those names. The Default theme is used when no colour theme is specified in the preferences. The Print theme is used for printing. The standard version of the Print theme uses a printer-friendly set of colours, using shades of grey only, with fairly high levels of contrast. By saving your own theme with the name Print, you effectively change how printed keyboard layouts will look.

### Set 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 is defined by a KCAP resource, which is defined by Apple, and made available in a number of different locations in the system, depending on the version of the operating system. All of the KCAP resources that are in the system are available for use in Ukelele.

When you select this item from the menu, you get the dialog box in the figure. The pop-up button on the left lists all the distinct kinds of keyboards available on the system. When you select one, a brief description is given in the box below.



Name	Description
Original Mac	Original Mac. Can't be used for OS X 10.2 or later
Original Mac with keypad	Original Mac with keypad. Can't be used for OS X 10.2 or later
Mac Plus	Mac Plus. Obsolete, not supported by any Mac OS X hardware, now used for unknown third party keyboard
Extended ADB	Extended ADB keyboard. Note that ADB keyboards cannot be used with OS X 10.3 and later
Standard ADB	Standard ADB keyboard. Note that ADB keyboards cannot be used with OS X 10.3 and later
Portable ADB	Portable ADB keyboard. Note that ADB keyboards cannot be used with OS X 10.3 and later
ADB keyboard II	ADB keyboard II. Note that ADB keyboards cannot be used with OS X 10.3 and later
PowerBook ADB	PowerBook ADB keyboard. Note that ADB keyboards cannot be used with OS X 10.3 and later
Adjustable keypad	Keypad for an Apple Adjustable Keyboard
Adjustable	Apple Adjustable Keyboard
PowerBook Extended	PowerBook Extended keyboard, with function keys and inverted T arrow keys
PowerBook Subnotebook	PowerBook Subnotebook keyboard, with function keys and inverted T arrow keys
PowerBook embedded keypad	PowerBook keyboard with embedded keypad and inverted T arrow keys (including Titanium PowerBook)
Original USB (Cosmo)	Original USB (Cosmo) keyboard
1999 Japanese PowerBook	1999 Japanese PowerBook
USB Pro keyboard	USB Pro keyboard
PowerBook/iBook, 2nd cmd key	PowerBook/iBook 2001 keyboard, with 2nd command key and a different arrangement of function keys (brightness, sound, etc)
USB Pro with F16	USB Pro keyboard with F16 key
Pro with F16	Pro keyboard with F16 key
PS2 keyboard	PS2 keyboard
PowerBook USB internal	PowerBook USB-based internal keyboard
Third party	Third party keyboard
Aluminium Wireless	Aluminium Apple Wireless Keyboard, released with Aluminium iMac, 2007. Also used for recent MacBook, MacBook Pro and MacBook Air models.
Aluminium Apple	Aluminium Apple Keyboard, released with Aluminium iMac, 2007
MacBook (Late 2007)	MacBook (Late 2007) keyboard, with Fn key but no embedded keypad

As of Mac OS X 10.7.4, the current list is in the table on the opposite page.

In the top right of the dialog, there is a second 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.

In most cases, you will also see a number of entries at the bottom of the list without descriptions, such as "Unnamed keyboard (ID = 22)". Some are useful, but on some systems you'll find one with ID 6000 which doesn't actually have any keys defined in it. If you choose that one, you'll get a message that the keyboard couldn't be used, and Ukelele will go back to the last keyboard used (i.e. the one you were using before you chose this menu item), or a default keyboard if that couldn't be used for some reason.

### **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.

#### **7.2.6. 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.

#### **7.2.7. Help menu**

Access to the Ukelele help book is through this menu.

## 8. Troubleshooting

---

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

### 8.1. 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. The file name must have the extension .keylayout, which is automatically provided by Ukelele. Or if it is in a bundle, the bundle must have the correct format and be in the correct folder. Bundles created by Ukelele should have the correct format.

If the keyboard layout is in the correct folder, have you logged out and logged in again? 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.

Check again the relevant part of System Preferences. You're looking for the Language & Text (International on Leopard or earlier) pane, and the Input Sources (Input Menu on Leopard or earlier) 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 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.

If you cannot find your keyboard layout in the Input Sources (Input Menu) tab of Language & Text (International) 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.

## **8.2. I edited the keyboard layout, but the changes don't seem to work**

If you edit a keyboard layout that is already installed into one of the Keyboard Layouts folders, then logging out and logging in again will not be sufficient to make the system load the new version. What happens is that, when you log in, the system looks at when the *folder* was last modified when deciding whether to scan and load new keyboard layouts. The solution is to drag the keyboard layout file to the desktop, then put it back into the Keyboard Layouts folder, and then log out and log in again. That should ensure that the changed version is loaded.

## **8.3. 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.

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.

## **8.4. 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 ID... command from the Keyboard menu, then choose Unicode from the pop-up list, and put 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. 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.

### **8.5. 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 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.

### **8.6. 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.

### **8.7. 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), 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 (Language & Text on Snow Leopard or later) 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.

### **8.8. 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 <sup>^</sup>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 <sup>^</sup>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 (⌘, Unicode U+05D0) for the ‘a’ key, then you would want to change the key binding from “<sup>^</sup>a” (meaning control-a) to “<sup>^</sup>\U5D0”. If you are creating your own local key bindings dictionary, you would make a binding like “<sup>^</sup>\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!



## 9. 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 30th March 2012.

<a href="https://groups.google.com/forum/?fromgroups#!forum/ukelele-users">https://groups.google.com/forum/?fromgroups#!forum/ukelele-users</a>	User group home page
<a href="http://www.unicode.org">http://www.unicode.org</a>	Details of the Unicode standard.
<a href="http://developer.apple.com/technotes/tn2002/tn2056.html">http://developer.apple.com/technotes/tn2002/tn2056.html</a>	Apple's technical note describing the XML format for keyboard layout files.
<a href="http://scripts.sil.org/IWS-TOC">http://scripts.sil.org/IWS-TOC</a>	SIL's Non-Roman Script Initiative's book, "Implementing Writing Systems".
<a href="http://www.unibuc.ro/en/cd_sorpaliga_en">http://www.unibuc.ro/en/cd_sorpaliga_en</a>	Sorin Paliga's home page (in English), with various keyboard layouts available for download.
<a href="http://www.evertype.com/celtscript/celt-keys.html">http://www.evertype.com/celtscript/celt-keys.html</a>	Celtic keyboard layouts from Everson Typography.
<a href="http://www.xenotypetech.com/">http://www.xenotypetech.com/</a>	Assorted Unicode language kits from XenoType Technologies.
<a href="http://www.bekkoame.ne.jp/~n-iyana/researchTools/asianextended.html">http://www.bekkoame.ne.jp/~n-iyana/researchTools/asianextended.html</a>	Nobumi Inayaga's Asian Extended keyboard layout.
<a href="http://www.ynlc.ca/languages/font/index.html">http://www.ynlc.ca/languages/font/index.html</a>	Yukon keyboard layout from the Yukon Native Language Centre.
<a href="http://apagreekkeys.org/">http://apagreekkeys.org/</a>	GreekKeys, a commercial package for polytonic Greek.

<a href="http://homepage.mac.com/herr/">http://homepage.mac.com/herr/</a>	Vietnamese keyboard layouts by Gero Herrman
<a href="http://ntresources.com/unicode.htm#Macintosh">http://ntresources.com/unicode.htm#Macintosh</a>	Polytonic Greek keyboard layout by Rodney Decker
<a href="http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&amp;item_id=ipa-sil_keyboard">http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&amp;item_id=ipa-sil_keyboard</a>	IPA keyboard for the SIL-IPA fonts
<a href="http://www.alanwood.net/unicode/fonts_macosx.html">http://www.alanwood.net/unicode/fonts_macosx.html</a>	Unicode fonts and links to specific keyboard layouts
<a href="http://www.linguistsoftware.com/lhebu.htm">http://www.linguistsoftware.com/lhebu.htm</a>	Linguist Software's Laser Hebrew font and keyboard layout
<a href="http://www.linguistsoftware.com/lgku.htm">http://www.linguistsoftware.com/lgku.htm</a>	Linguist Software's Laser Greek font and keyboard layout. They also have other keyboard layouts.
<a href="http://gandalf.aksis.uib.no/mufi/keyboards/macOSXkeyboard.html">http://gandalf.aksis.uib.no/mufi/keyboards/macOSXkeyboard.html</a>	Medieval Unicode Font Initiative keyboard layout
<a href="http://www.thlib.org/tools/">http://www.thlib.org/tools/</a>	Nepali fonts and keyboard layouts from the Tibetan & Himalayan Digital Library (you will need to navigate to Fonts & Related Issues, then to Nepali Fonts)
<a href="http://www.redlers.com/download.html#Anchor-Keyboard-11481">http://www.redlers.com/download.html#Anchor-Keyboard-11481</a>	Various keyboard layouts linked to by RedleX, makers of Mellel, a Unicode-savvy word processor
<a href="http://www.madanpuraskar.org/downloads.php">http://www.madanpuraskar.org/downloads.php</a>	Nepali Unicode Romanized keyboard layout from Madan Puraskar Pustakalaya
<a href="http://www.ukij.org/mac/">http://www.ukij.org/mac/</a>	Links to Uyghur software, including keyboard layouts
<a href="http://www.aatseel.org/macintosh_cyrillic">http://www.aatseel.org/macintosh_cyrillic</a>	Cyrillic fonts and keyboard layouts
<a href="http://www.cs.helsinki.fi/u/kkuloves/dvorak.shtml">http://www.cs.helsinki.fi/u/kkuloves/dvorak.shtml</a>	Dvorak layout for Finnish by Kimmo Kulovesi
<a href="http://ebmp.org/p_downlds.php">http://ebmp.org/p_downlds.php</a>	Unicode keyboard layouts from the Early Buddhist Manuscripts Project (you want EasyUnicode)
<a href="http://www.unirioja.es/cu/jvarona/TeXkeylayout.html">http://www.unirioja.es/cu/jvarona/TeXkeylayout.html</a>	Keyboard layouts for typing TeX/LaTeX by Juan L. Varona

<a href="http://homepage.mac.com/WebObjects/FileSharing.wa/wa/default?user=shah&amp;templatefn=FileSharing5.html&amp;xmlfn=TKDocument.5.xml&amp;sitetfn=TKSite.3.xml&amp;aff=consume&amp;cty=US&amp;lang=en">http://homepage.mac.com/WebObjects/FileSharing.wa/wa/default?user=shah&amp;templatefn=FileSharing5.html&amp;xmlfn=TKDocument.5.xml&amp;sitetfn=TKSite.3.xml&amp;aff=consume&amp;cty=US&amp;lang=en</a>	Malaysian Jawi keyboard layout by Ahmad Sahar
<a href="http://homepage.mac.com/thgewecke/tckbs.html">http://homepage.mac.com/thgewecke/tckbs.html</a>	Various keyboard layouts by Tom Gewecke
<a href="http://homepage.mac.com/thgewecke/pkeyboards.html">http://homepage.mac.com/thgewecke/pkeyboards.html</a>	Pointers for users of PC keyboards, also by Tom Gewecke
<a href="http://m10lmac.blogspot.com/">http://m10lmac.blogspot.com/</a>	Tom Gewecke's blog page
<a href="http://sites.google.com/site/macmalayalam/">http://sites.google.com/site/macmalayalam/</a>	Fonts and keyboard layout for typing in Malayalam
<a href="http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&amp;item_id=MacHGTransUniKey">http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&amp;item_id=MacHGTransUniKey</a>	Keyboard layout for transliterated Greek and Hebrew, developed by Joan Wardell
<a href="http://www.hcs.harvard.edu/~jrus/Site/cocoa-text.html">http://www.hcs.harvard.edu/~jrus/Site/cocoa-text.html</a>	Information on key bindings and the Cocoa text system.
<a href="http://belkadan.com/blog/2012/04/Keyboard-Adventures/">http://belkadan.com/blog/2012/04/Keyboard-Adventures/</a>	Information about how the “intended language” system works with OS X and how to do more with the “press and hold” system.
<a href="http://en.wikipedia.org/wiki/IETF_language_tag">http://en.wikipedia.org/wiki/IETF_language_tag</a>	Wikipedia article about language tags used for the “intended language” system.
<a href="http://apple.stackexchange.com/questions/20505/add-characters-to-the-press-and-hold-character-picker-in-os-x-lion/44928#44928">http://apple.stackexchange.com/questions/20505/add-characters-to-the-press-and-hold-character-picker-in-os-x-lion/44928#44928</a>	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.



## 10. 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 10.7.4.

Language	Code	Notes
Arabic	ar	Arabic script
Bulgarian	bg	Cyrillic script
Catalan	ca	Also used for Valencian
Cherokee	chr	
Chinese	zh	zh-Hans (Simplified), zh-Hant (Traditional)
Croatian	hr	Latin script
Czech	cs	
Danish	da	
Dutch	nl	Also nl-BE (Belgium), i.e. Flemish
English	en	Also en-AU (Australian), 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	
Korean	ko	
Latvian	lv	
Lithuanian	lt	

Language	Code	Notes
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
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	
Ukrainian	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 ã ä å ã ā</string>
    <key>Strings</key>
    <string>a à á â ã ä å ã 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 å ã ā</string>
    <key>Strings</key>
    <string>à â ª a æ á ä å ã ā</string>
</dict>
```

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