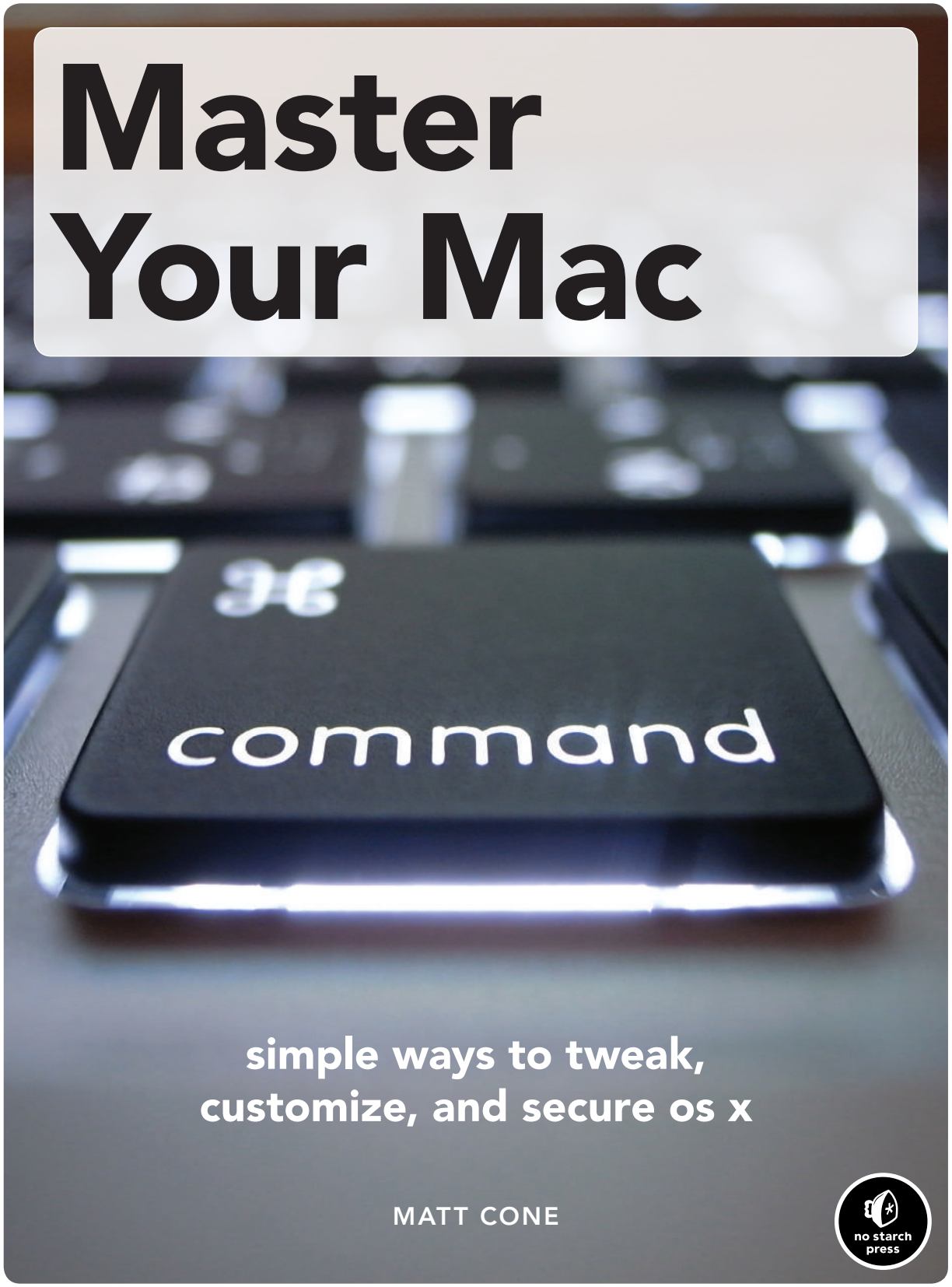
Master Your Mac



simple ways to tweak, customize, and secure os x

MATT CONE

no starch press

13 Creating a Bluetooth Proximity Monitor

Ready to get up, stretch, and walk away from your computer? First you'll have to follow a checklist that has probably become second nature—pause iTunes, mute the volume, set your instant messenger applications to away, and lock the screen. It all had to be performed manually, until now.

The next time you step away from your computer, how about just taking your iPhone, smartphone, or iPad with you? With a *Bluetooth proximity monitor* running, your Mac will know the device has moved out of range and will run an AppleScript to perform a series of actions, like locking the screen and pausing iTunes. When you return, your Mac will run a different AppleScript to unlock your screen and start playing music again.

Since you'll be creating the AppleScripts in the project, you can make your Mac do

anything you want when the proximity of the Bluetooth device changes. And you don't need an iPhone or iPad—any Bluetooth-enabled device can work with the Proximity application, including smartphones running the Android operating system.

Project goal: Turn your Mac into a proximity monitor capable of performing actions when a Bluetooth device moves into or out of range.

What You'll Be Using

To automatically tell your Mac when you step away and what to do when you leave, you'll use the following:

Proximity (*http://code.google.com/p/reduxcomputing-proximity/*, free)

AppleScript Editor

How Bluetooth Proximity Detection Works

Macs are capable of detecting Bluetooth devices up to 10 meters (33 feet) away. The Proximity application takes this factoid and makes it useful. By using your Mac's internal Bluetooth sensor, Proximity can run AppleScripts when the state of a connected Bluetooth device changes. There are two states for a device: in range and out of range. Proximity can run separate AppleScripts for each state, making this a great way to automatically perform a variety of actions on the Mac when you walk away from it and when you come back.

Setting Up the Bluetooth Proximity Monitor

If AppleScripts are the real substance of your Bluetooth proximity monitor, Proximity (*http://code.google.com/p/reduxcomputing-proximity/*, free) is the glue that holds everything together. You need to install and configure this application before you create the scripts that will be triggered as you go away and return.

Here's how to configure Proximity:

1. Open the Proximity application. The Proximity icon appears in the menu bar.
2. From the **Proximity** menu, select **Preferences**. The window shown in Figure 13-1 appears.
3. In the first field, enter a time in seconds to specify the interval at which Proximity will check for the Bluetooth device. This number should be as low as possible for fast detection and script execution. You don't want to get back to your desk and find that Proximity hasn't detected your device leaving yet!
4. Click **Change Device**. The Detect Bluetooth Device window appears.
5. Select a Bluetooth device from the list and then click **Select**. If you're trying to select an iPhone or iPad, you may need to open the Bluetooth settings to make it discoverable. (In iOS, tap **Settings**, then **General**, and then **Bluetooth**.)

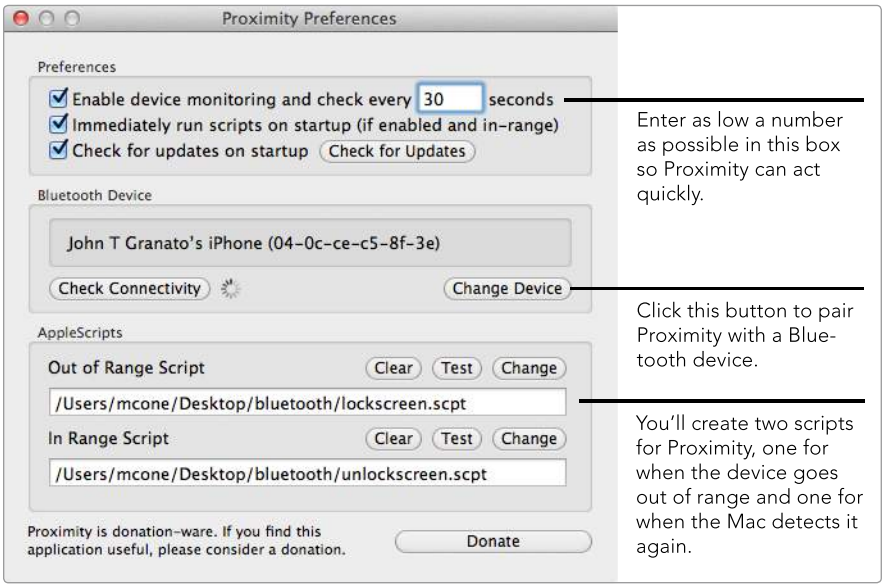


FIGURE 13-1: *Configure Proximity's preferences for maximum effectiveness.*

After you create the scripts, you'll need to circle back around to Proximity to select them in the preferences.

Performing Actions Based on Bluetooth Proximity

Here comes the fun part. Open the AppleScript Editor application (it's in the Utilities folder) and create two new scripts—an *Out of Range* script, which will be triggered when you walk away, and an *In Range* script, which will be triggered when you return. Then start adding some of the code snippets provided below.

Of course, you don't have to add all of the code provided below, nor do you have to limit yourself to it. You can mix and match any of the actions you want—or add new actions of your own—to create custom scripts. Whatever you put in your scripts, it's a good idea to test them to make sure they do what you expect before adding them to Proximity.

Locking the Screen

One of the most obvious uses for Proximity is security related—it can lock the screen when you walk away and unlock it when you return. Protecting your computer from unauthorized access is a best practice, but sometimes it's hard to remember to start the screensaver before you walk away. Implementing this as part of a Bluetooth proximity monitor automatically engages this security control.

Put this in the *Out of Range.scpt* script to lock the screen (you or someone else can still unlock it without the Bluetooth device in proximity, but doing so requires a password):

```
tell application "System Events"
     tell security preferences
          set require password to wake to true
     end tell
end tell
try
     tell application "ScreenSaverEngine"
          activate
     end tell
end try
```

Now put this in *In Range.scpt* to unlock the screen when you return:

```
tell application "System Events"
     tell security preferences
          set require password to wake to false
     end tell
end tell

tell application "ScreenSaverEngine" to quit
```

Now screen locking is automated. Just remember to take your iPhone (or whatever Bluetooth device you're using) with you when you walk away from the computer, or your Mac will never know you've left.

Pausing iTunes

Pausing and resuming iTunes is another useful feature you can bake into your proximity monitor. When you add this code snippet, iTunes pauses your music when you walk away and then resumes playing when you come back.

Put this in *Out of Range.scpt* to pause playback:

```
tell application "iTunes" to pause
```

And then put this in *In Range.scpt* to resume iTunes when you return:

```
tell application "iTunes" to play
```

Pausing and resuming playback is just one example of how you can control iTunes with AppleScript. iTunes has excellent AppleScript support, so use your imagination to add other features. For example, you could tell iTunes that when you return, it should start playing a random album, download and play an hourly news podcast, or even change the iTunes equalizer settings. For more ideas, visit a website called Doug's AppleScripts for iTunes (*http://dougscripts.com/itunes/*).

Setting an Away Message

If you use Messages or Adium (*http://adium.im/*, free) for instant messaging, you can automatically display an away message when you leave your desk to let your friends know that you're not there.

Put this in *Out of Range.scpt* to set both Messages and Adium to "away" status:

```
tell application "Messages"
     set status to away
end tell
tell application "Adium" to go away with message "Out of Bluetooth range"
```

The out-of-range script now has two commands—one for Messages and one for Adium. Obviously, you'll want to replace *Out of Bluetooth range* with the Adium away message of your choice.

Put this in *In Range.scpt* to change your status in Messages and Adium to "available":

```
tell application "Messages"
     set status to available
end tell
tell application "Adium" to go available
```

Walking away from your computer is now a quick way to end an annoying conversation, just as in real life!

Adding the Scripts and Testing Everything

Now that you have the scripts, it's time to select them in Proximity. Here's how:

1. From the **Proximity** menu, select **Preferences**. The window shown earlier in Figure 13-1 appears.
2. To select an *Out of Range Script*, click **Change** and select the AppleScript file you want to run.
3. To select an *In Range Script*, click **Change** and select the appropriate AppleScript file.
4. You can test the scripts by clicking **Test**. If the scripts don't work the way you expect, make changes and test them again.

Now it's time for the real test. Try turning off the Bluetooth device. When your computer recognizes that the device has disappeared, Proximity will execute the out-of-range script. Then turn the Bluetooth device back on—Proximity will execute the in-range script. If Proximity doesn't work as quickly as you were expecting, try lowering the time setting in Proximity's preferences.

Additional Ideas for Creating a Bluetooth Proximity Monitor

You might be wondering what happens in a worst-case scenario. If you go to lunch and lose your iPhone, what's going to happen when you get back to your desk? Well, Proximity won't be able to trigger the in-range script, but you'll still be able to deactivate your screensaver with your password. So it's no big deal if you lose your Bluetooth device—this proximity stuff isn't very serious security. For practical security advice, see Part 6.

The possibilities for a Bluetooth proximity monitor are virtually endless. You're limited by your scripting abilities, but even if you aren't a scripting whiz kid, you can still do quite a bit. For example, you can download some scripts off the Mac OS X Hints website (*http://hints.macworld.com/article.php?story=201107190007230*) to create an alarm for your laptop with an application called iAlertU (*http://sourceforge.net/projects/ialertu/*, free). Just install the scripts and then walk away from your computer with your Bluetooth device. Your computer will be protected by iAlertU—if anybody messes with your Mac, it will start emitting a loud alarm.

Home automation is another intriguing possibility. If you positioned your Mac close enough to your garage or front door and connected it to a home automation app like Indigo (*http://www.perceptiveautomation.com/indigo/*, $$$), you could do things like turn a television off when you leave the house and turn the lights on when you get home.