CHAPTER 3

Before You Get Started

Okay, now that you know the ins and outs of Bluetooth, you’re probably eager to find out how to integrate Bluetooth with Java. Well, this chapter is all about doing just that. But wait! Before you learn about how to use Bluetooth and Java, you need to know when it is not a good idea to use the two technologies together.

When NOT to Use Bluetooth and Java

You should not use Bluetooth with Java for the following purposes:

- Signal strength indicator
- Voice applications
- Distance measuring

The next few sections explain why to avoid those scenarios.

Signal Strength Indicator

Let’s say that you have two Bluetooth units, and you want to know what the signal strength is between them. A good example is when you want to use the services of a network access point. A signal strength indicator would let you know if you were within range. Well, Java is not the ideal language for that sort of application because that kind of information is not exposed to the level where a JVM would have access to it. The JVM will let you know if you are within range or not within range; there is no middle ground. In this scenario, you’re better off using a native language for your device such as C or C++.

Voice Applications

Now, you’ve already read Chapter 2, and you realize that Bluetooth is a really great technology because you have the ability to transmit voice and data
information wirelessly to other Bluetooth devices. Suddenly, you get ideas bubbling in your head about how great it would be to create a speech-to-text application on your Bluetooth-enabled phone. Unfortunately, Java (especially J2ME) is not well suited to this arena just yet. Performance is a key factor in voice-based applications, and once again, in this case, you're better off using a native language such as C. However, this application may be feasible to do in Java if the Java Real-Time Technology can be incorporated.

**Distance Measuring**

The best wireless technology for accurately measuring distance is light waves and not radio signals. Light waves are direct, and the calculations can be pretty simple because the speed of light (in various mediums) is pretty well documented. Using radio signals to measure distance can be quite tricky, and one of the best ways to do that is to use triangulation, like GPS transceivers do. Whether you are using Java or C, Bluetooth might be a viable technology for triangulation, but definitely not for calculating or measuring accurate distances.

**NOTE** The key word here is accurate. You can definitely use Bluetooth for proximity measurement (i.e., where in the building is Bruce Hopkins?). In fact, the Ericsson BlipNet does exactly that! See Chapter 11 for more information on the Ericsson BlipNet.

So, to put it succinctly, you can only do what is possible using the constraints of the Bluetooth technology and what the JVM exposes to you. If the JVM only gives you access to the RFCOMM layer for communication, then you're stuck with it. If the OBEX layer is not exposed to the JVM, then don't expect to be able to send objects. To increase application portability, your Java Bluetooth vendor should implement the Java Bluetooth specification created through the JCP.

**Understanding the JCP**

The JCP is the Java Community Process, and it is the formal procedure to get an idea from a simple concept incorporated into the Java standard. This process allows developers and industry experts to shape the future of the Java standard. Popular APIs like Java USB, Java Real-Time, Java Printing, Java New I/O, J2ME MIDP 1.0, J2ME MIDP 2.0, JDBC 3.0, EJB 2.0, and even JDK 1.4 all went through