



JAVA & BLUETOOTH – THE FUTURE STARTS NOW

BY KARL MCCABE, CTO ROCOCO SOFTWARE

Bluetooth is being embedded in many different device types – phones, PDAs, printers, cars and fridges, to name a few. To create applications that take advantage of the increasing number of Bluetooth devices on the market, three attributes in particular need to be addressed: *application portability*, *over-the-air (OTA) provisioning* and *ease of programming*. Java provides these key features and opens the door to a new wave of Bluetooth application possibilities.

In order for Bluetooth devices to form useful collaborative networks, it's essential to have applications that run across many platforms. Currently, each Bluetooth stack exposes its own proprietary API on one or more Operating Systems (OS). If a developer writes an application for one OS-stack combination, that application will not work with other combinations. This is a key barrier to the emergence of independent application development for Bluetooth devices. To achieve this level of portability it's necessary to have both a standard Bluetooth API and a software environment that works with any OS – in short, a standard Java API.

A second important attribute is Over-The-Air Provisioning, which refers to one's ability to download an application over a wireless link into a device, and then run that application on that device. This kind of code mobility is central to any software framework that is more than "just" a browser. It means a user can load applications into their device on the fly. These could be Bluetooth applications downloaded over a 3G link, for example. Or the application could be provisioned over a Bluetooth link. For example, a user might download an application from a Bluetooth access point, use the application and then simply delete it when they're finished. Virtual tourist guides, interactive maps of public spaces and feature-rich conference assistants are just some of the application possibilities. *Code mobility is central to the Java platform* – Java applications are interpreted by Java Virtual Machines (VM) and are represented in a standard 'bytecode' format. Any device that has a VM can interpret and run Java bytecode. Applications can now be shared across Bluetooth links, as well as simply sending application data across these links.

For Bluetooth to become the default choice for peer-to-peer application programmers, it must be easy to use. Many abstractions have been developed over the years for developers building complex distributed systems – from Remote Procedure Calls through to Web Services. Each abstraction aims to allow the programmer to concentrate on the application problem at hand, rather than on low-level connectivity. This level of abstraction is often missing from Bluetooth stacks – they require a detailed level of programming that will become unsuitable as the technology becomes more mainstream. Java provides a framework that can be co-opted for this purpose – the Generic Connection Framework in the Java 2 Platform, Micro Edition is a set of ease-of-use abstractions for network programming.

As an illustration of the capabilities of Java as a software framework for Bluetooth applications, consider a user who wants to print the contact list from their mobile phone. The user walks up to a Bluetooth-enabled printer. The phone and the printer are not "pre-configured" to work with each other – the phone software does not include a device driver for the printer. However, using Java and Bluetooth, the phone can discover the printer, discover the print service and then request the Java driver for the printer. The driver is provisioned over-the-air into the phone, which now has the ability to print to this device-type. Using the code mobility and portability of Java, two devices can come together and collaborate to achieve some useful task.

In March, 2002, the world's first standard API for Bluetooth – the Java APIs for Bluetooth Wireless Technology (JABWT) version 1.0a - was approved. The JSR-82 Expert Group developed this specification as part of the Java Community Process. The group was chaired by Motorola and comprised 18 companies, including Rococo Software. The specification builds on top of the Connected Limited Device Configuration (CLDC) component of the Java 2 Platform, Micro Edition (J2ME). It provides a set of Java



abstractions for each of the key Bluetooth host stack components – L2CAP, RFCOMM, SDP, OBEX and key HCI functionality – and it supports the foundational Bluetooth profiles GAP, SPP, GOEP and SDAP.

In the next months Java/Bluetooth products will begin to appear on the market. For example, Rococo Software has implemented the JABWT standard in its Impronto product range, including a set of developer tools that allows the Java Bluetooth developer to quickly become productive and to build compelling, easy-to-use applications. For more information about the JABWT standard, check out www.jabwt.com, an online resource for Java and Bluetooth sponsored by Rococo Software.