

TIME FOR A BREW

.....▶

How does Java work with Bluetooth?

In fact, does it provide a suitable development platform for development and deployment of Bluetooth solutions?

Java specialists Rococo Software explain

rococoTM

Sun's Java technology has particular attributes (portability, mobility, internet-ready) that make it compelling for wireless application development and deployment. Key vendors, such as Motorola, Siemens and Palm, are endorsing the language and the Java 2 Platform Micro Edition (J2ME) environment, for their next generation mobile devices. Certain key emerging technologies, such as JXTA, Jini and JavaSpaces, are driving the Java platform forward, and leading the way for standards-based application development in the wireless, short-range peer-to-peer (P2P) and ad hoc networking arenas. As a leading short-range wireless technology, Bluetooth is well positioned to underpin the next generation of P2P and ad hoc networking infrastructure and applications. Here, we examine the convergence of Java and Bluetooth, particularly in Sun Microsystems' Java Community Process, and the benefits that accrue from this convergence.

WIRELESS JAVA

Sun has released a specific variant of the Java environment called J2ME (Java 2 Platform, Micro Edition). J2ME is targeted specifically at consumer devices and embedded devices. It consists of a set of configurations and profiles (not to be confused with Bluetooth profiles). A J2ME Configuration defines the minimum set of Java class libraries and virtual machine features supported on a particular category of devices. The Connected Limited Device Configuration (CLDC) is the most relevant one for Bluetooth applications, as it targets mobile devices with small memory budgets (160KB to 512KB) and wireless, potentially intermittent connectivity. It comprises a fast, small footprint virtual machine (the KVM) and a stripped down but fully functional Java API subset. A J2ME Profile is targeted at application developers. It is layered on top of a configuration and is the set of APIs available on a particular family of devices. Configurations target horizontal market segments, whereas profiles target vertical segments. One important profile is the Mobile Information Device Profile (MIDP), which provides a set of User Interface components, a persistence mechanism and a HTTP connection capability for use in mobile phones, PDAs and other handheld mobile devices.

STANDARD JAVA BLUETOOTH APIS

Java APIs are standardised under the auspices of the Java Community Process (JCP), which

issues Java Specification Requests (JSRs). If accepted, a JSR Expert Group is formed from industry leading experts, and is tasked with providing both the standard itself and, importantly, a reference implementation of the standard. Standards emerging from such a process are, by implication, 'implementable'. The JSR-82 Expert Group is tasked with standardising Bluetooth APIs for the Java Platform. These APIs are targeted at CLDC, and will most likely manifest themselves as a J2ME Profile (see diagram). Rococo Software is a member of this Expert Group, which is chaired by Motorola. The current status of the standard will be presented to the Java community at the JavaOne show in San Francisco in June 2001—the specification will be finalised by December 2001.

THE J2ME BLUETOOTH PROFILE

There are many benefits of a standardised Java API for Bluetooth. Key among them are:

■ **Portability**

Compliant Java code can be ported quickly to run on different operating systems. This allows developers to develop a single application for multiple devices, with a minimum of code being changed from device to

device. In its best expression, it achieves something called 'write once, run anywhere'—a given application once written can run on anything that supports Java according to the standard.

■ **Mobility**

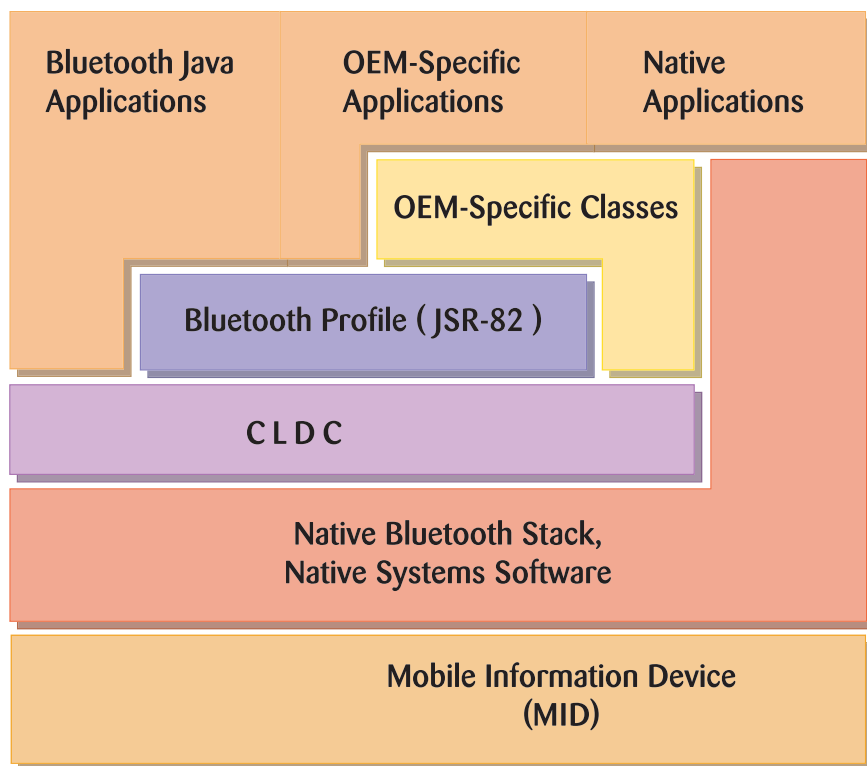
Java code can be dynamically delivered OTA (over-the-air) to mobile devices and then executed locally. This allows developers to develop applications that could be 'served' to a customer via a mobile network, a web portal, or via another customer ('pass it on'). The ability to load up new applications 'on-the-fly', perhaps disposing of them once used, is important for resource-constrained mobile devices.

■ **Net-enabled**

Java was designed with the internet in mind—it is one of the only languages 'plumbed' with internet support at its core.

■ **Developer support**

Java has the fastest growing support among developers worldwide. Current estimates are that around 2-2.5 million developers now use Java in their daily work. A seasoned Java programmer with-



out wireless experience can very quickly write programs for the mobile environment using Java on J2ME. A recent study by the Evans Data Corporation found that over 30 per cent of wireless developers were targeting the Java/J2ME platform.

■ **Bluetooth interoperability**

Effective interoperability between Bluetooth devices, currently a key goal for the Bluetooth community, depends on two things. Firstly, stacks must interoperate with each other at all levels of the stack from baseband upwards, and they must agree on protocol commands, events, actions and Protocol Data Units (PDUs). Secondly, applications that use these stacks must agree on certain aspects of their use—for example, what, if any, initialisation commands must be issued; what order dependencies, if any, are there between commands; what interpretation is given to data passed as parameters to commands. Though Bluetooth profiles address many of these concerns, it is only through a standard, semantically specified and agreed API specification that true interoperability may be achieved.

■ **New Developments—
JXTA, Jini, JavaSpaces**

Java is not just a language, it is a programming environment or platform. It strives to bring together important emerging technologies under one umbrella, making them available to a huge and growing set of Java programmers. This approach allows application developers to re-use best-of-breed patterns and design approaches when designing applications. Java platform additions—some recent, some more mature—such as JXTA, Jini and JavaSpaces will be key technologies for Bluetooth applications.

JXTA (pronounced ‘juxta’ as in ‘juxtapose’) was launched by Sun in April 2001. It is a programming and computing platform designed to solve a number of problems in the area of peer-to-peer networking. Readers familiar with the legal tribulations of Napster will know that peer-to-peer networking is, broadly speaking, a networking topology whereby end-user devices are connected directly to other end-user devices, without going through a third-party ‘server’, for the purposes of sharing

some information or services. Although many such networks have emerged, no interoperable, platform independent platform has existed on which to build these collaborative applications. JXTA is such a platform. It is an open-source technology that addresses such issues as peer discovery, peer information sharing, peer membership, and asynchronous pipes. It proposes an XML-based approach to advertising peer resources. JXTA is neutral to the underlying transport, although it seems clear that short-range wireless technologies, such as Bluetooth could play a key role in its success. A device using JXTA over Bluetooth could discover other local JXTA devices and form ad hoc, peer-to-peer networks with them to achieve some common, collaborative goal, or share information or services.



Jini, officially launched in mid-2000, is a technology that may be used to federate groups of users and resources required by those users. It aims to allow a network to be more flexible and more easily administered, with resources being easily located by both

humans and computational clients. Jini provides framework components for managing federated services, such as a look-up service for locating services, a leasing capability for controlling use of services and security and transaction capabilities for coherent inter-service interactions. Jini operates at a level of abstraction that makes it attractive to application developers needing to solve an application problem, rather than worry about underlying networking issues. It provides a paradigm that may be useful to Bluetooth applications, since it allows services to be loosely coupled—that is, Jini nodes and services may appear and disappear from a network as a natural consequence of its operation. This loose coupling of entities is an important quality in an environment of mobile devices moving in and out of range of each other.

JavaSpaces is a Jini Service. It provides the abstraction of a JavaSpace into which components, known as ‘entries’ can be put. These entries may be found, and they may be looked up by performing a ‘read’ (passive) or a ‘take’ (destructive), and they may be written to a space. It includes a notification mechanism to allow an application to be notified when an entry is written that matches some criteria. This form of abstraction is particularly useful for passing information from stage to stage in a workflow model, or for applications where redemption of some token is important, e.g. redeeming a voucher with a service provider. Interestingly, JavaSpaces also provides a useful abstraction for layering over actual, physical spaces. In a Bluetooth access point scenario, this may be useful as an application model for mapping zones, or spaces, in which access points are installed, and for specifying which services are available in which zones.

We believe that Java and its associated technologies will have a pivotal role to play in facilitating the development and deployment of Bluetooth applications.

ABOUT THE AUTHOR

Karl McCabe is the CTO and founder of Rococo Software. Formerly senior engineering manager at IONA Technologies, with responsibility for IONA’s Java product lines, he brings a deep knowledge and understanding of distributed systems to the sphere of wireless ad hoc networking. He may be contacted at karl.mccabe@rococosoft.com.