

Written by Gary M. Katz

# Understanding JTWI

WILL MANUFACTURERS AND CARRIERS FINALLY GET THE JAVA TECHNOLOGY PLATFORM NEEDED TO POWER THEIR NEXT-GENERATION HANDSETS?

*Java Technology for the Wireless Industry (JTWI) will have a huge impact on Java applications, enabling them to have much greater functionality. Here we look at some of the capabilities we can expect to see in the next generation of Java for mobile phones.*

Introduction by Bill Ray, WBT's Editor-in-Chief

**J**ava was supposed to offer *Write Once, Run Anywhere*, but anyone involved in developing applications for the mobile industry knows well that this maxim is true of only the most basic applications. Inconsistencies in implementations, and proprietary extensions, mean that every application has to be tweaked for each device it runs on, or pay the price in usability and speed. As an example, consider sound: MIDP 1.0 has no capability to make noise, which is something of a drawback when trying to create a mobile gaming experience. Thus every manufacturer – and device – has its own API for making noises, meaning that no game will run on two different devices.

But mobile Java has come a long way, and with Java Technology for the Wireless Industry (JTWI; also known as JSR 185) new standards are required and possible inconsistencies addressed. Support for JTWI has been enormous across the business, with carriers, device manufacturers, and even customers embracing the standard as quickly as they can, eager to make real the dream Java has offered for so long, but realized so rarely. Even while the official tests are incomplete, the first devices are on the market, and JTWI looks well placed to supplant other Java standards in record time. We asked Applix to run us through what developers can expect of the first Java standard to really deserve the name.

Java Emerges As a Dominant Handset Platform

Readers of this publication are almost certainly familiar with the rapid evolution of Java technology in mobile phones. Java 2 Platform Micro Edition (J2ME) is one of Sun's biggest success stories in recent years. The numbers are impressive. According to Sun, its mobile Java technology is incorporated in more than 120 million handsets worldwide; used by 70 mobile carriers worldwide; and shipped in more than 200 different handset models from 27 manufacturers worldwide. Industry analyst firm ARC Group estimates that worldwide shipments will exceed one billion units before 2010. To achieve this degree of market penetration, J2ME has succeeded in gaining acceptance with the two crucial players in the wireless industry: carriers and device manufacturers. How?

Wireless carriers are motivated to launch compelling Java services that will generate new revenue streams. Java technology enables them to offer consumers powerful applications to extend the functionality of mobile phones, and to develop new revenue streams from the services these new applications enable.

Handset manufacturers like the idea of a common application layer. It enables them to migrate from native applications developed for a specific handset to a suite of built-in applications that can ship with any of their current and future phones. Java technology provides an excellent means to differentiate and extend the fea-

ture sets of manufacturers' mobile phones – helping them improve their standing with key wireless operators by delivering more compelling, better-selling handsets.

To meet the requirements of wireless operators and handset manufacturers for a more optimal Java platform for next-generation mobile phones, new specifications, such as Mobile Information Device Profile (MIDP) 2.0 and Java Technology for the Wireless Industry (JTWI), have been approved in the Java Community Process (JCP).

New Standards Help Java Technology Keep Its Lead in Mobile Services Deployment

Java technology has a significant head-start on its competition, primarily BREW and .NET, in providing the dominant platform for mobile services. When compared to the applications and services being deployed today, the earliest incarnation of J2ME, MIDP 1.0, provided very basic functionality. In particular, in MIDP 1.0 some fundamental networking and security support was not specified, and the GUI library was very limited. Optional extensions needed to be added to MIDP 1.0 phones to provide expanded functionality, such as the ability to play sounds, manipulate images, or flash the phone's backlight.

These limitations led to the definition of MIDP 2.0 and later, JTWI. These Java Specification Requests (JSRs) were approved in the JCP and are expected to become fundamental components of next-generation Java technology-enabled mobile phones.

At the time of its announcement in the fall of 2002, MIDP 2.0 was considered a big step forward for J2ME since it opened the door to a wide array of useful functionality

that would enable developers to build the next generation of mobile phones. Among these functional improvements were an enhanced user interface, multimedia and game functionality, greater connectivity, over-the-air provisioning, and end-to-end security to mobile information devices such as mobile phones and entry-level PDAs. Mobile phones supporting MIDP 2.0 have recently arrived on the market from leading handset manufacturers such as Motorola.

### JTWI: A Platform for Future Generations of Cellphones

JSR-185, more popularly known as JTWI, promises to dramatically expand the capabilities of a new generation of mobile phones. JTWI was defined in the JCP by an expert group of leading mobile device manufacturers, wireless carriers, and software vendors, and describes how various JSRs associated with MIDP 2.0 – such as Wireless Messaging API (WMA) and Mobile Media API (MMAPI) – work together to form a complete handset solution for the wireless services industry.

JSR-185 provides three high-level deliverables:

1. A road map of mobile phone-related JSRs and a description of their availability in different markets around the world.
2. A specification describing the essential client components of an end-to-end mobile phone environment and the recommended combinations of J2ME technologies. JSR-185 specifies requirements that work to enhance end-to-end compatibility.
3. An integrated Reference Implementation (RI) and Technology Compatibility Kit (TCK) for the technologies described in the specification.

In late July 2003, JSR-185 left the custody of the JCP as a fully defined specification with an RI and a TCK. Release 1 is now officially in the hands of carriers, handset manufacturers, and software developers, so now is an opportune moment for mobile Java developers to explore what enhancements it offers them and prepare accordingly.

JTWI comprises essential, complementary technologies that unlock critical features for developers and operators by guaranteeing the presence of specific APIs. Two categories of JSRs are defined in JTWI: mandatory and conditionally required.

Devices claiming JTWI compliance must pass the associated TCK for all mandatory JSRs and must also pass the associated TCK for conditionally required JSRs if the device contains a capability referenced by the JSR and exposes that functionality to Java applications (e.g., a phone that exposes multimedia APIs, such as a media player, must use JSR-135 and pass the associated TCK).

### Mandatory JSRs

Following is a summary of the mandatory and conditionally required JSRs that comprise JTWI:

- **CLDC 1.0 (JSR-30):** The JTWI specification is designed to be implemented on top of Connected Limited Device Configuration (CLDC) 1.0, which provides basic API services and the virtual machine itself.
- **MIDP 2.0 (JSR-118):** MIDP provides a standard set of Java APIs that provide core application functionality required by mobile applications, such as the user interface, local data storage, network connectivity, and application life cycle management. MIDP 2.0 greatly expands the functionality of MIDP 1.0 by providing gaming, audio, security and network extensions, improved high-level user interface support, and support for MIDlets (Java applications) that can wake up in response to a trigger (in the form of a “push” event) that the phone receives from a server.
- **Wireless Messaging API (JSR-120):** Wireless messaging is a key service that can be used by games, as well as business and commerce applications. WMA pro-

vides platform-independent access to wireless communication resources, enabling developers to build intelligent, connected Java applications that can send and receive short messages, and provide access to network-specific services, such as Short Message Service (SMS) and Unstructured Supplementary Services Data (USSD) used with GSM networks. WMA is designed to foster interoperability between different manufacturers and networks – a big benefit for consumers who wish to exchange data with friends and associates on a different network. Better multiplayer games are enabled through this JSR (e.g., chess, battleship, etc.), as well as new business and commerce applications (e.g., broadcast-type applications such as location-based services, restaurant finders, news, and weather updates).

### Conditionally Required JSRs

- **Mobile Media API (JSR-135):** MMAPI is an optional component of JTWI that is required if media services are exposed through Java APIs. An exciting feature of MMAPI is its ability to support capture of audio data with a microphone or still or video images with a camera. This enables consumers to record and send these memories to friends and relatives via wireless messaging. MMAPI also makes possible advanced multimedia games, such as driving simulators and combat games, where consistent visual feedback in the form of video clips and audio enhancements such as music, vibration, and sound effects are supported.

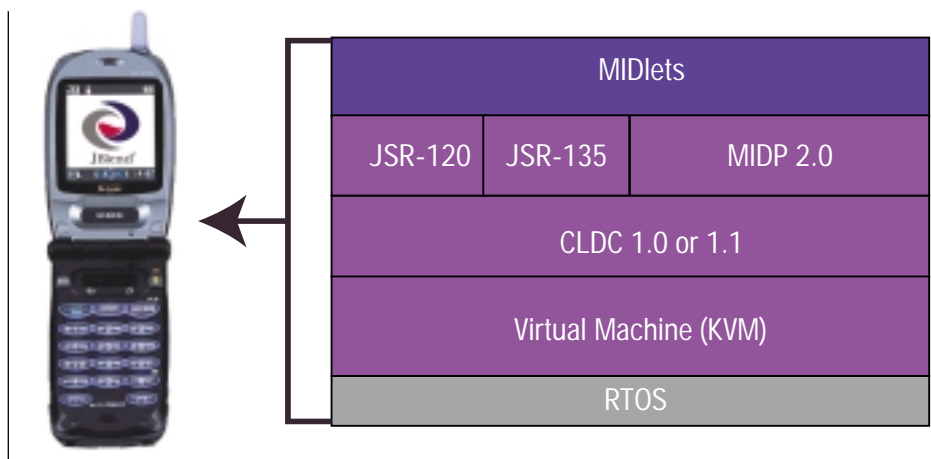


Fig. 1: Java software stack with JTWI components

- **CLDC 1.1 (JSR-139):** The major new functionality to CLDC 1.1 is support for floating point. If floating point capabilities are exposed through Java APIs, then CLDC 1.1 is required instead of CLDC 1.0.

Figure 1 illustrates a Java software stack implemented with the mandatory and conditionally required components of JTWI. JTWI also improves portability by clarifying existing specifications and defining key implementation details. For a summary of these clarifications, see Jonathan Knudsen's article, "Understanding JSR 185" (<http://developers.sun.com/techtopics/mobility/midp/articles/jtwi>).

### Looking Ahead – JTWI Release 2 and Beyond

Future releases of JTWI may continue to "roll-up" other JSRs that have direct relevance to the needs of vendors in the wireless services industry. The following JSRs were under consideration for inclusion in Release 2 and future JSR-185/JTWI roadmaps at the time the JTWI Roadmap was published in January 2003:

**JSR-75:** PIM APIs

**JSR-82:** Bluetooth

**JSR-164/165:** SIMPLE presence and IM

**JSR-172:** Web services

**JSR-177:** Security APIs

**JSR-179:** Location services

**JSR-180:** SIP

**JSR-184:** 3D APIs

**JSR-186/187:** Generic presence and IM

**JSR-190:** Events

### Implementing JTWI on Wireless Handsets

Implementing JTWI on mobile phones is not trivial. It requires expertise and experience with CLDC 1.0/1.1, MIDP 2.0, WMA, and MMAPI at a minimum. Deployment is even more complicated if other JSRs will be incorporated into JTWI-compliant handsets. As of late November 2003, there were more than 25 different JSRs targeted at handsets. Surprisingly, manufacturers that license Java technology and develop their own Java implementations have not begun to question whether in-house development still makes economic sense. A key question to

ponder is: Do the costs of licensing, training, and staffing a dedicated development team and other associated R&D costs exceed the profit expected to be realized by the handset? If so, then outsourcing this J2ME development – including the JTWI component – to a Java platform partner may be a wise choice.

Manufacturers that have yet to implement Java technology are in an even more precarious position. While the long-term benefits of migrating to the Java platform may be compelling, and perhaps even essential for their survival, newcomers to Java technology will need to find shortcuts to get their Java technology-enabled handsets to market quickly and successfully. They won't have the luxury of time to become experts in a set of new skills and technologies that are evolving at a breakneck pace. In these cases, turning to a Java platform partner may be their best option.

If a handset manufacturer is ready to outsource J2ME development, including JTWI, to a Java platform partner, the following criteria should be considered:

- **Deployment experience:** Experience bringing final designs to market as commercially successful products.
- **Technical experience:** Experience with key operating environments, operator-specific profiles and JSRs, including helping customers to pass the associated TCKs.
- **Integration experience:** Experience integrating the Java solution with ancillary hardware and software such as Java hardware acceleration, browsers, graphics engines, and streaming media players.
- **Time-to-market issues:** Complexity involved in implementing a third-party Java solution into the existing phone hardware and software architecture. A key question is: Can the Java solution be quickly ported and integrated to ensure key deadlines are met or, better, create competitive time-to-market advantages?
- **Java community clout:** Strength of relationship with Sun Microsystems and other JSR specification lead companies, and involvement in the Java Community Process, particularly JSR expert groups.

- **Java technology solution:** Attributes of the partner's Java solution. Several key questions need to be considered regarding the implementation itself: Is it well optimized and compliant with the appropriate specifications? Does it provide excellent performance (using software and/or hardware acceleration, as required); efficient memory management and small footprint; robustness and security; and support for a wide variety of operating systems, chipsets, and operator profiles? Is it built on original Sun source code to ensure compatibility? Does it provide support for K native interface (KNI) to enable handset manufacturer native class libraries such as dialers, phone books, and graphics to be called?

JTWI certainly contains all the ingredients essential for some very compelling phones. For manufacturers and carriers alike, the recipe for the success of these phones will be bulletproof implementations of the JSR-185 specification. Working with an experienced Java platform partner may be the best way to ensure that the project is a rousing success.

### References

- **Sun Microsystems Press Release:** [www.sun.com/smi/Press/sunflash/2003-10/sunflash.20031014.4.html](http://www.sun.com/smi/Press/sunflash/2003-10/sunflash.20031014.4.html)
- **JTWI/JSR-185 Road Map, Sun Microsystems:** [http://jcp.org/aboutJava/communityprocess/jsr/JSR185\\_roadmap.pdf](http://jcp.org/aboutJava/communityprocess/jsr/JSR185_roadmap.pdf)
- **JSR 185: Java Technology for the Wireless Industry:** <http://jcp.org/en/jsr/detail?id=185>
- **Understanding JSR-185 by Jonathan Knudsen:** <http://developers.sun.com/techtopics/mobility/midp/articles/jtwi>
- **The Java Community Process:** <http://jcp.org/en/home/index>
- **A complete list of all JSRs:** <http://jcp.org/en/jsr/all>

### About the Author

Gary M. Katz is director of corporate marketing for Aplix Corporation, a leading provider of Java technology for mobile phones and other consumer electronics products.

[gary@aplixcorp.com](mailto:gary@aplixcorp.com)