

Symbian on Java

Author: Martin de Jode
Status: Version 2.05
Date: March 2004

1. Introduction

In this paper we describe Symbian's Java technology. We shall look at the evolution of Java on Symbian OS, updating the story to include additions to the platform in the latest release of Symbian OS, Version 8.0. We shall then briefly discuss Symbian OS phones and how these relate to the development of Java on Symbian OS. In the final section we shall consider Symbian's Java strategy and provide a glimpse as to how the Java platform may evolve in the future.

However, first we shall discuss the importance of Java technology from the perspective of Symbian and its stakeholders.

2. Why Java?

In a recent [press release](#) Sun Microsystems announced that 250 million mobile phones supporting Java technology have shipped worldwide. Furthermore, this figure is set to increase. A report from the ARC Group ("Future Mobile Handsets", 2002) predicts that there will be more than 600 million Java capable handsets in 2005. As a consequence, Operators and ISVs will often be more interested in the hundreds of millions of phones supporting MIDP than in native applications targeted at a specific OS, particularly for mass-market services and applications that must be delivered to a range of different mobile phones.

The attractions of Java as a development environment were recognized early on by Symbian, which made a strategic decision to provide a first-class Java runtime environment to supplement the native C++ environment. This opened up Symbian OS phones to the 3 million Java developers.

Of course, Java doesn't provide all the answers. The mass-market, cross-platform appeal comes at a price in terms of performance and functionality compared with native applications. However, the Java "tax" is becoming less onerous, and the benefits more compelling, as the gap between Java and native performance closes and Java functionality is enhanced.

3. Evolution of Java on Symbian OS

Symbian's first Java implementation, based on Sun's JDK 1.1.4, was released as part of Symbian OS v5 in 1999.

For their next release Symbian decided to take advantage of the reduced memory footprint offered by PersonalJava (compared to the burgeoning JDK) and used the PersonalJava 1.1.1 specification (essentially a subset of the JDK 1.1.6 API set) as the basis for their Java implementation. This release, Symbian OS Version 6.0, became available in 2000. Symbian OS v6.0 also provided an implementation of Sun's JavaPhone API, a vertical extension to the PersonalJava platform, providing access to the underlying phone functionality including the ability to:

- access telephony functionality
- send and receive datagrams
- manipulate address book and calendar information.

In the meantime Sun was revising the Java strategy. In 1999, acknowledging that “one size doesn’t fit all”, Sun announced the splitting of Java into three versions:

- Java 2 Enterprise Edition (J2EE)
- Java 2 Standard Edition (J2SE)
- Java 2 Micro Edition (J2ME)

(Java had been re-branded Java 2 with the release of the JDK 1.2).

The Enterprise Edition was aimed at providing end-to-end enterprise solutions focusing on the server side, while the Standard Edition targeted the desktop environment. The Micro Edition was targeted at a range of consumer and embedded electronic devices with constrained resources and it is this we will concentrate on. Since the remit of the Micro Edition covers a wide range of devices in various market segments, the Micro Edition itself is subdivided into Configurations targeted at particular hardware configurations. The most appropriate configuration for mobile phones is the Connected, Limited Device Configuration CLDC, (160-512 KB memory available for Java, battery powered, slow, possibly intermittent, connection). Sitting on top of the CLDC is the Mobile Information Device Profile (MIDP) which specifies an API set appropriate for mobile information devices such as phones.

Soon it was clear that J2ME MIDP was gaining momentum in the wireless space as phone manufacturers endorsed the idea of a lightweight Java environment suitable for mass-market phones. Symbian recognised the strength of the MIDP movement by including J2ME MIDP 1.0 as its standard Java offering in Version 7.0 of Symbian OS, released in 2002, as well as back-porting it to earlier versions. However, it was also recognised that the MIDP 1.0 specification severely limited the scope of MIDlets. As a consequence the richer, but larger memory footprint PersonalJava/JavaPhone was retained as an option available to licensees. (For a discussion of the differences between the PersonalJava/JavaPhone and J2ME MIDP development environments read the [earlier paper](#)..

J2ME has progressed a long way since its first conception in 1999. Although MIDP 1.0 generated considerable enthusiasm amongst the Wireless Java Community, it was realised that MIDP 1.0 on its own provided limited access, and so limited capabilities, to the functionality of the typical smartphone from within a MIDlet. Consequently, soon after the release of MIDP 1.0 the wireless community started working on enhancing the capabilities of MIDP. This has been manifested in the MIDP 2.0 Java Specification Request (JSR 118), released in its final form in November 2002 and a range of extension API JSRs, all forming part of the Java Community Process. These optional packages increase the functionality available to MIDlets.

In 2003 Symbian released Version 7.0s of Symbian OS. This introduced support for J2ME MIDP 2.0, which brings a new, finer-grained security model, enhanced UI API, Game and audio APIs and the Push Registry to the Java platform. In addition, Symbian OS v7.0s provides an implementation of the Java API for Bluetooth Wireless Technology (JABWT, JSR82), giving MIDlets access to the Bluetooth stack and the Wireless Messaging API (WMA, JSR 120), allowing MIDlets to send and receive SMS messages. To ensure “Best in Class” performance Version 7.0s also makes use of Sun’s high performance CLDC HI VM.

Nokia has used Symbian OS v7.0s for Version 2.0 of their Series 60 Developer Platform. In addition to the functionality that comes as standard in Version 7.0s (see above), the Series 60 Developer Platform 2.0 also provides an implementation of the Mobile Media API (MMA, JSR 135) providing Java support for video playback, tone generation and photo capture, supplementing the audio API that comes as part of MIDP 2.0.

The latest version of Symbian OS, Version 8.0, was publicly announced in Q1 2004. This enhances the J2ME CLDC/MIDP implementation adding the following optional packages to Symbian OS: Mobile Media API (JSR 125), Mobile 3D Graphics (JSR 184), File GCF (part of JSR 75) all running on top of Sun’s CLDC HI 1.1 Virtual Machine (VM). In addition, the Java implementation is now fully compliant with the Java Technology for the Wireless Industry specification (JTWI, JSR 185). The JTWI is an initiative defined via the JCP to specify a minimum set of APIs and behaviour that a compliant phone should support. By targeting the JTWI, ISVs and 3rd party developers can know that their applications will run on the largest possible number of phones. Release 1 of the specification mandates MIDP 2.0, CLDC 1.0 and WMA as a minimum API set with the MMA also required if multimedia functionality is exposed to Java. Symbian OS v8.0 also integrates support for the Universal Emulator Interface

(UEI) allowing Symbian MIDP emulators to fully integrate with standard tools such as Sun's Wireless Toolkit and IDEs such as JBuilder and Sun One Studio.

4. Java on Symbian OS phones

In this section we will try to relate phones to Symbian OS versions. Current Symbian OS devices can be broadly categorized into four groups based on UI:

- NTT DoCoMo FOMA
- Nokia 9200 Series Communicator
- Series 60
- UIQ

The first group includes the Symbian OS-based FOMA F2051, F2102V and F900i 3G phones. These support Java for i-Mode (DoJa). DoJa is a NTT DoCoMo proprietary profile that runs on top of CLDC.

The Nokia 9200 Communicator series have been in the market place for some time. This device family was based on Symbian OS v6.0 and therefore comes with PersonalJava and the JavaPhone API. A MIDP 1.0 environment is also available for this series from [Forum Nokia](#).



Nokia 9200 Communicator



Sony Ericsson P800



Sony Ericsson P900

Figure 1 Some Symbian OS Phones supporting PersonalJava

The Nokia 7650 smartphone, released in Q3 2002, is a member of the Nokia Series 60 UI family. This model was the first Symbian OS v6.1-based Series 60 Developer Platform 1.0 device and supports Java MIDP 1.0. In addition, Nokia provide a couple of proprietary extension APIs (in the com.nokia.mid.sound and com.nokia.mid.ui packages). The Nokia 3650 is another Series 60 phone which shipped in Q1 2003. This device also runs MIDP 1.0, but additionally adds in support for the WMA and the MMA. Variants of this device include the Nokia 3620 and 3660 models. Other Series 60 Developer Platform 1.0 devices include the Nokia N-Gage gaming console (Q4 2003), the Siemens SX1 (Q1 2004) and Sendo X (coming soon).



Nokia 7650

Sony Ericsson P800

Nokia 3650

Nokia N-Gage

Figure 2 Some MIDP 1.0 enabled Symbian OS phones

The Sony Ericsson P800, released in Q4 2002, is based on Symbian OS version 7.0 and uses the UIQ 2.0 user interface. This device supports MIDP 1.0 and PersonalJava. The 3G Motorola A920/A925 are also based on UIQ 2.0 (the A920 being released in Q3 2003) and support similar Java capabilities to the P800.

With the arrival of Symbian OS Version 7.0s Symbian's Java technology brought in support for MIDP 2.0. Nokia have used v7.0s as the basis for their Series 60 Developer Platform Version 2.0. The Nokia 6600 is based on this platform and became the first Symbian OS MIDP 2.0-enabled device to ship (Q4 2003). In addition to MIDP 2.0, the Nokia 6600 provides the WMA, MMA and JABWT optional packages running on top of Sun's CLDC HI VM. Other Series 60 Developer Platform 2.0 devices in the pipeline include the Nokia 6620 and the Panasonic X700.

The new Java technology encompassed in v7.0s was back-ported to Symbian's UIQ reference design in version UIQ 2.1. The Sony Ericsson P900 was the first phone based on UIQ 2.1 and shipped in Q4 2003. This also supports MIDP 2.0 plus the WMA and JABWT J2ME packages and, for the diehards, maintains support for PersonalJava. UIQ 2.1-based phones in the pipeline include the BenQ P30 and Motorola A1000.



Nokia 6600

Sony Ericsson P900

BenQ P30

Panasonic X700

Figure 3 Some Symbian OS MIDP 2.0 enabled phones

The Java specification of the main developer platforms supported by Symbian OS phones currently available in Western markets is summarised in Figure 4 below.

Developer Platform	OS Version	PersonalJava	MIDP	J2ME optional packages		
				WMA	JABWT	MMA

Series 60 v 2.x	v 7.0s	no	2.0	yes	yes	yes
UIQ 2.1	v 7.0 ²	yes	2.0	yes	yes	no
UIQ 2.0	v 7.0	yes	1.0	no	no	no
Series 60 v 1.x	v 6.1 ¹	no	1.0	Yes ³	no	Yes ³
9200 Series	v 6.0	yes	1.0	no	no	no

1. Includes back-port of some v7.0 technology
2. Includes back-port of some v7.0s technology
3. Depending on platform minor version number

Figure 4 Java specification for current Developer Platforms

Symbian OS v8.0-based licensee products will follow in due course.

5. Symbian and Java – the next steps

Symbian’s Java strategy is enabling the emerging market for advanced consumer services. Future releases of Symbian OS will implement Java APIs for location, web services, PIM, Bluetooth push, security and trust and other technologies that will allow developers to create and run larger, more interesting, games, applications and services. Additionally, Symbian is also represented on the majority of the expert groups of J2ME JSRs.

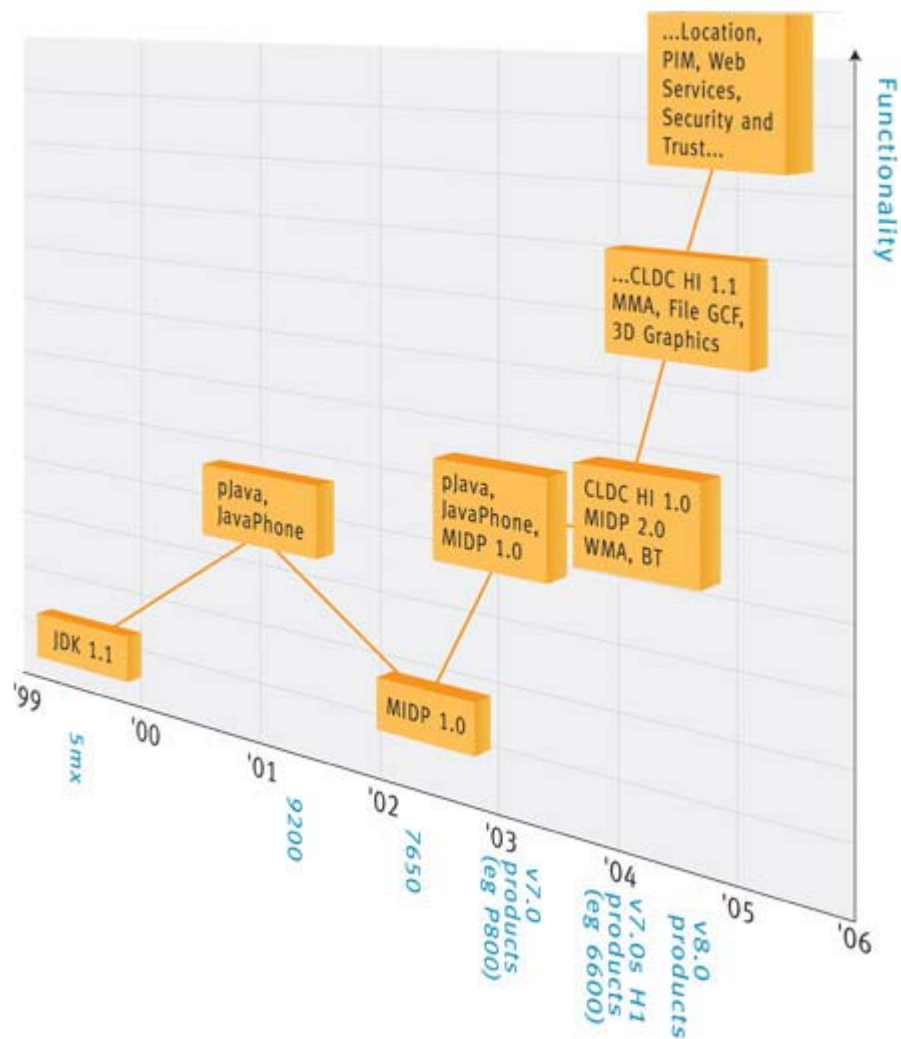


Figure 5 Matching functionality to market needs

Symbian believes that, for the foreseeable future, CLDC provides the right Java technology base for mass market requirements, consumer in particular. However, in the short term Enterprise clients need the additional features and functionality available in CDC/Personal Profile and therefore an implementation is available for Symbian OS for those handset manufacturers that require it. For instance, the recently announced Nokia 9500 not only supports Symbian's CLDC based implementation (including MIDP 2.0, WMA, MMA and JABWT) but also includes IBM's J9 CDC-based Java implementation providing support for the Personal Profile.

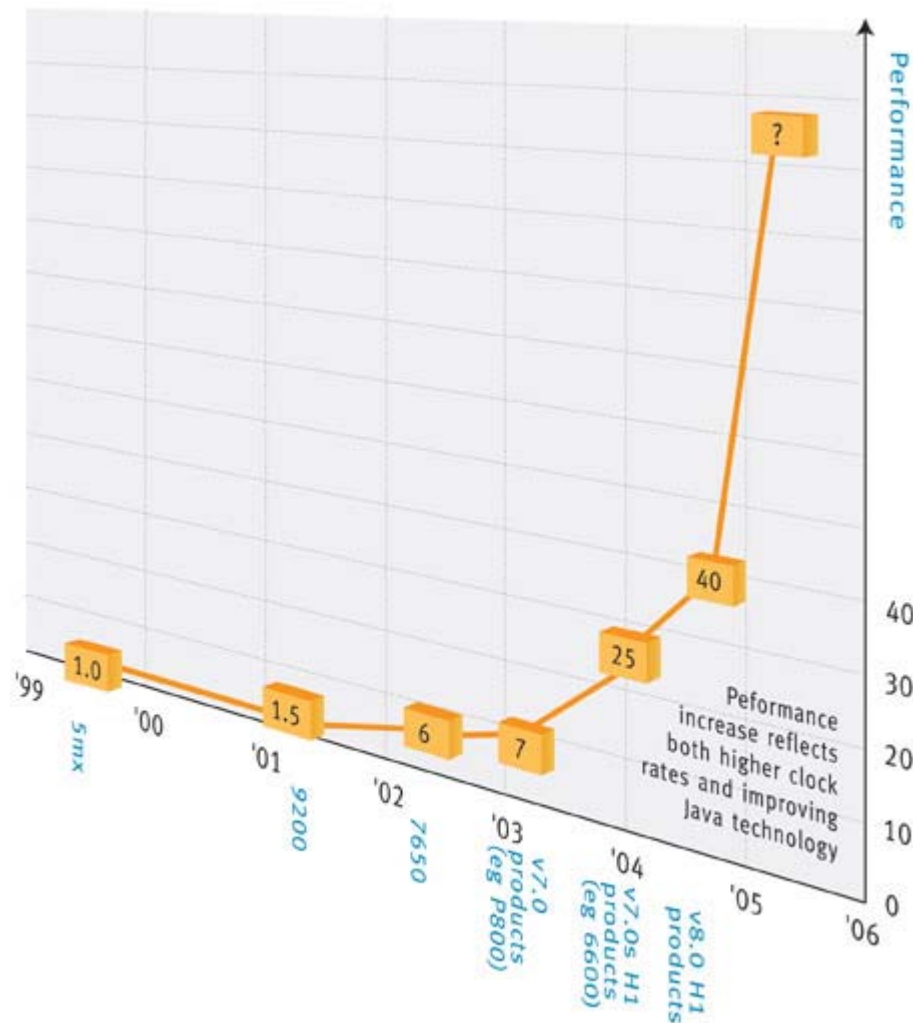


Figure 6 Symbian OS Java performance

Java performance (per MHz) has steadily improved (see Figure 6), due to technologies such as dynamic adaptive compilation and hardware acceleration of byte code interpreters, and to optimisations in graphics, native interface, and embedded performance. At the same time clock rates are increasing. Thus, future v8.0 based mobile phones are likely to deliver around 40 times the performance of the original Symbian OS v5 JDK1.1 implementation (this is using Symbian's CLDC benchmark, a more realistic indicator than the embedded Caffeine benchmark). Such a level of performance means we can run complex advanced data services, that go beyond simple "screen scraping" applications and wireless services. These advanced services will make use of local resources for storage, processing, and communication (e.g. Bluetooth), only synchronizing with other clients and back end data sources as and when needed.

6. Benefits of Java on Symbian OS

Developers, network operators, and service providers will benefit from the added value provided by Java on Symbian OS:

- Symbian's Java implementation is robust. It is running on the Symbian OS kernel, which is itself designed to operate reliably. User data loss in Symbian OS phones is very rare, and system resets are rarely required.
- Symbian OS Java is fast. Symbian's MIDP 1.0 provides support for ARM's VTK software accelerated interpreter, whilst MIDP 2.0 uses Sun's CLDC HI VM. Symbian's Java UI components map directly to native UI components. Symbian OS in turn is extremely efficient.
- The Java implementation has a small footprint, taking advantage of Symbian OS's lean and mean philosophy.
- Symbian's implementation of new APIs derived from the Java Community Process is extending the functionality of MIDlets.

With the performance and capabilities of wireless Java on Symbian OS continually improving it now offers ISVs and service providers a viable development environment for targeting the mass-market for advanced data services in the wireless space.

7. Glossary

CDC	Connected Device Configuration Defines a Java runtime environment for "high end" consumer devices with constrained hardware resources such as set top boxes, PDAs and Communicators. Falls under the J2ME umbrella
CLDC	Connected, Limited Device Configuration Defines a Java runtime environment for "low end" devices with highly constrained hardware resources such as mobile phones and pagers. Falls under the J2ME umbrella
ISV	Independent Software Vendor
JABWT	Java API for Bluetooth Wireless Technology An optional API that allows J2ME applications to access Bluetooth functionality
JCP	Java Community Process An open organisation of Java developers and licensees whose charter is to develop and revise Java technology specifications, reference implementations, and technology compatibility kits
JSR	Java Specification Request The process by which new Java specifications are defined. Part of the Java Community Process.
JTWI	Java Technology for the Wireless Industry An initiative to provide: <ul style="list-style-type: none"> • A roadmap of mobile phone related JSRs, indicating their availability in different markets around the world • A specification describing the essential client components of an end-to-end wireless environment • Provides a Reference Implementation of the technology and a Technology Compatibility Kit
J2ME	Java 2 Micro Edition A version of Java aimed at consumer and embedded devices including cell phones
MIDlet	An application written for the Mobile Information device profile (MIDP)

- MIDP** **Mobile Information Device Profile**
Vertical extension of CLDC. Defines an API set for Mobile information devices such as cell phones

- MMA** **Mobile Media API**
An optional API providing a high level interface to sound and multimedia capabilities

- UEI** **Universal Emulator Interface**
Interface enabling integration of emulators with tools such as the Wireless Toolkit

- WMA** **Wireless Messaging API**
An optional API providing access to wireless messaging resources including SMS