Special Article on Advanced i-mode Mobile Phones

New Technology Report

Implementing Java in Mobile Phones

Eriko Ooseki and Kazuhiro Yamada

The 503i series is a lineup of Java-enabled mobile phones geared to games and agent services. Java has attracted a great deal of attention due to its "Write Once, Run Anywhere" feature.
This article reviews the Java functions of the 503i series.

1. Introduction

Functions of the conventional i-mode mobile phones included speech telephony, content browsing, e-mail exchange and melody downloading. The latest series can download and execute programs in addition to these functions, owing to the Java★ runtime environment [1], [2]. This article describes the Java runtime environment installed in the i-mode mobile phones, and explains what has been made possible in the environment.

2. Java Runtime Environment

Generally speaking, the term Java has two meanings: the programming language itself and its runtime environment. The latter is installed in the 503i series, so that the mobile phones can run Java programs.

Ordinary (non-Java) programs are translated into machine language dependant on the type of the operating system (OS), which is a process called compilation. As they can only run on a specific type of OS, a separate program is required to make them run on another OS. A solution to this is the Virtual Machine (VM) model, in which Java programs are converted into byte codes that can be processed by the VM on any OS in the course of compilation. As long as the VM is installed, Java programs can be run on any OS. **Figure 1** illustrates how it works.

Recently, much attention has been paid to Java 2 Platform

★ Java: An object-oriented programming language suited for use on networks, developed and advocated by Sun Microsystems in the U.S.

Micro Edition (J2ME), which is a version of Java designed for small electronic devices. VM is expected to be installed in a wide range of electronic products, from mobile electronic devices like Personal Digital Assistants (PDA) and mobile phones, to consumer electronics and automobiles. J2ME/CLDC (Connected Limited Device Configuration) is installed in the 503i mobile phones.

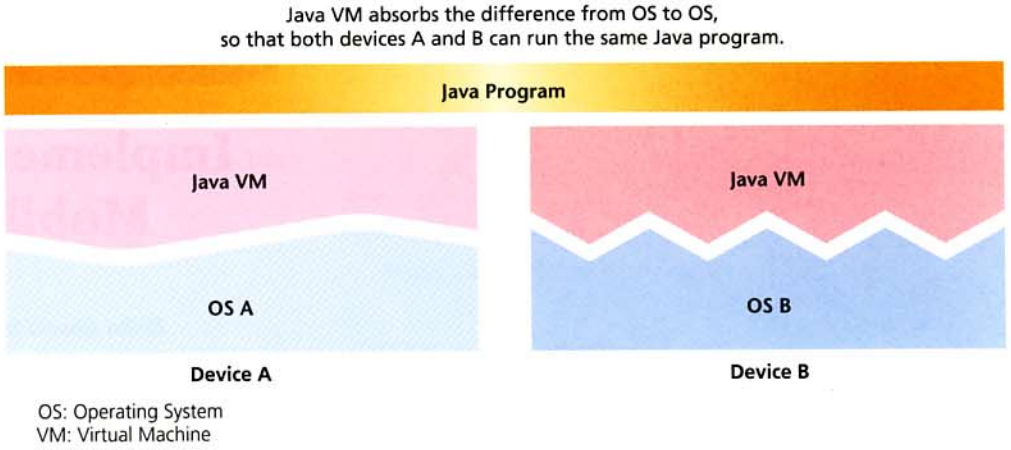


Figure 1 How Java Works

In the 503i series, Java programs run as applications rather than applets that are run from the browser. Java programs run by themselves in 503i. The browser is used only when the Java application is being downloaded. Once downloaded, the Java program can be run without activating the browser, meaning that Java programs are just as user-friendly as the existing i-melody and i-anime, as far as downloading and storing are concerned.

3. CLDC

CLDC is a configuration based on the K Virtual Machine (KVM), which is designed for small, network-connected devices with limited memory capacity and inferior Central Processing Unit (CPU) performance. Basically, CLDC is a subset of Java 2 Standard Edition (J2SE), which is an upper Java runtime environment, aimed at facilitating upward compatibility and portability. CLDC does not support floating-point operations or reflections due to the VM's limited memory size.

For further information on CLDC, visit the website of Sun Microsystems in the U.S. [1].

4. i-mode Extended API

J2ME consists of the configuration of devices with similar basic performance requirements by category, and the profile defined for each specific field and industry type. The i-mode extended Application Programming Interface (API) belongs to the latter. Profiles standardized by the Java Community Process (JCP), in which DoCoMo is involved, include the Mobile Information Device Profile for the J2ME Platform (MIDP). wide range of devices are under the scope of MIDP, including interactive pagers, PDAs and mobile phones. The i-mode extended API is not compatible with MIDP, as the former has been optimized especially for i-mode services.

Figure 2 illustrates the architecture of the Java runtime environment in the 503i series. As shown in the Figure, models with a KVM/CLDC class library and an i-mode extended library are able to run the same programs. The i-mode extended library is specified to guarantee minimum compatibility. Certain types of applications that need to be programmed in consideration of the screen size and keypad characteristics, such as games, might require some fine-tuning, depending on the mobile phone. On the other hand, transaction-type programs (e.g., stock quotes, weather information) can be run without any modification. Each component is described below. As this article merely explains API in brief, refer to the "i-mode Java spec" disclosed at DoCoMo's website for further information [2].

4.1 Scratch Pad

A scratch pad is a space for storing data that is accessible from applications. For example, it is used for keeping the score of games and storing various parameters. In the 503i series, a scratch pad of 5-10 kbyte is available for use, provided that the size of the pad has been declared in the Application Descriptor File (ADF) before use, which is described later. For security purposes, the scratch pad cannot be shared by multiple applications.

4.2 User Interface

The user interface for running Java applications can be broadly divided into two types. One is the high-level API, which is based on components, and the other is the low-level

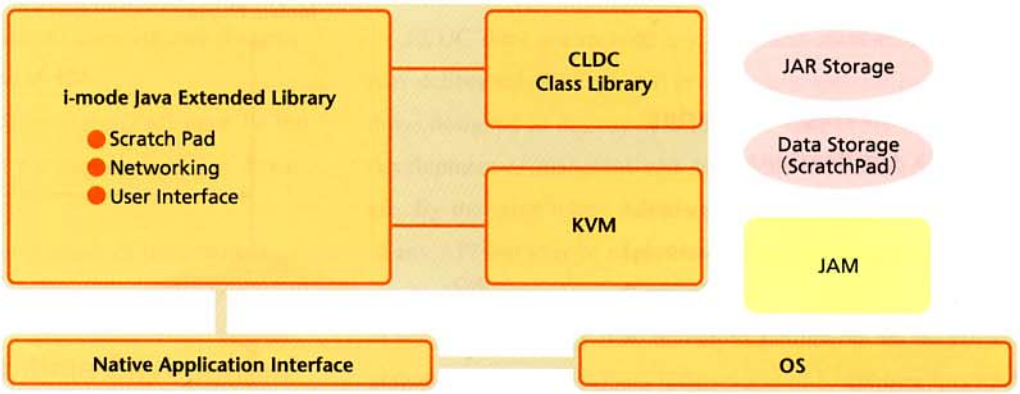
API, which is suitable for developing games and other applications.

Each component in the high-level API is designed in a fashion that makes it just as user-friendly as conventional browsers. **Table 1** shows the key components with reference to the browser display.

Application developers who use the components shown in Table 1 will acknowledge their inflexibility, but will be able to develop highly functional applications with a small number of codes.

With the use of the low-level API, application developers can take the mobile phone's screen size and keypad characteristics into account when programming, which is useful for developing applications like games. The low-level API allows much flexibility to application developers, but at the same time, it requires developers to manage all behavioral aspects of the application.

Other than API, the 503i series has a new element called the Java pictogram. (The "α" mark in **Photo 1**.) The pictogram appears on the screen when a Java application is running.



CLDC: Connected Limited Device Cofiguration
JAM: Java Application Manager
JAR: Java Archive
KVM: K Virtual Machine
OS: Operating System

Figure 2 i-mode Java Architecture

Table 1 Components and HTML

Component (Java)		HTML Tag (Browser)	Remarks
Text label		<PLAINTEXT>	
Image label		<IMG>	
Button		<INPUT type= "submit" >, etc.	
Text input		<INPUT type= "text" > <TEXTAREA>	Single-line text Multiple-line text
List	Option	<SELECT> <SELECT SIZE= "1" >	Pop-up type
	Single Selection	<SELECT SIZE= "number of lines" >	Box type
	Multiple Selection	<SELECT SIZE= "number of lines" MULTIPLE>	More than 1 line
	Check Box	<INPUT type= "checkbox" >	Box type
	Radio Button	<INPUT type= "radio" >	
Ticker		<MARQUEE>	

HTML: Hyper Text Markup Language

Photo 1 Java Pictogram

4.3 Network Usage

The Java runtime environment in the 503i series includes an API for Java applications to communicate with the server, which makes communication by Hyper Text Transfer Protocol (HTTP) and Hyper Text Transfer Protocol Security (HTTPS) possible. The mobile phones can thus acquire picture files and melody files via networks and playback them back while running a Java application. Moreover, the processing can be distributed between the mobile phones and the server by using the server's CGI, considering the 10 kbyte size constraint of Java Archive (JAR) files (applications that are downloadable to mobile phones) and the limited processing power of mobile phones. It should be noted that the use of networks must be declared in the ADF, which is described later.

5. Java Application Manager (JAM)

The 503i series is equipped with the Java Application Manager (JAM), which manages downloaded Java applications

and the VM. The key functions of JAM are as follows.

5.1 Downloading and Storing Java Applications

If there is an <OBJECT> tag in the HTML content, Java applications and JAR files are downloadable. **Figure 3** illustrates the download sequence of Java applications.

JAM acquires the ADF to determine whether the application is downloadable to the mobile phone. **Table 2** shows the information in ADF.

5.2 Management of Java Applications

As explained above, JAM determines whether the Java application is downloadable by referring to the information in the ADF. For example, if the JAR file turns out to be too large for the mobile phone to handle, the JAR file will not be downloaded. In addition to download, JAM manages many other tasks based on the set value of ADF. The details are described below.

(1) Auto Launching Setting

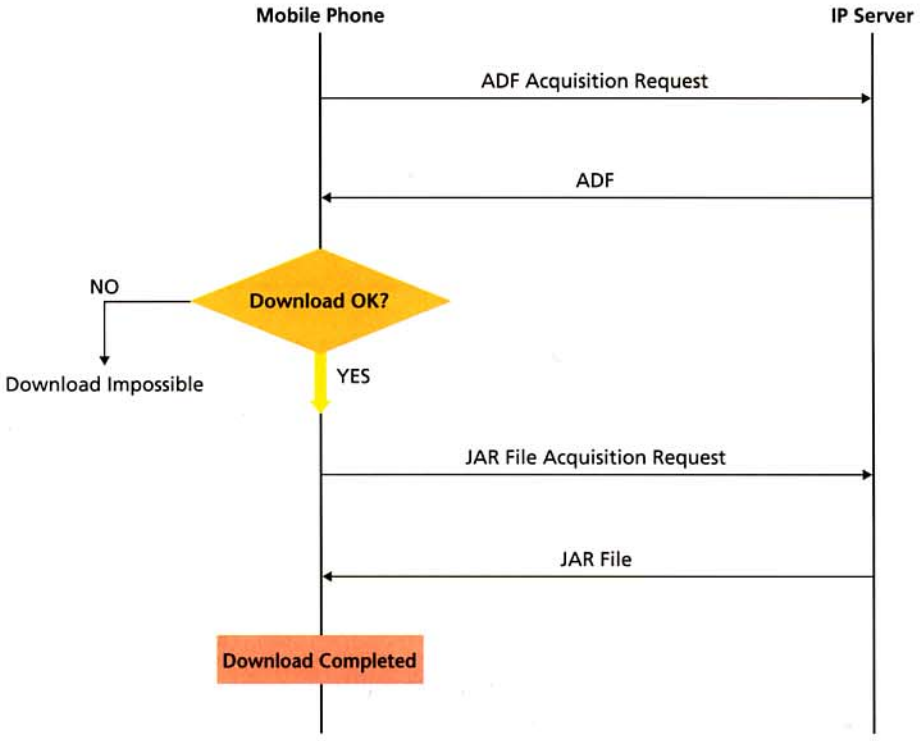
Java applications can automatically be launched at fixed intervals specified by the "LaunchAt" key in the ADF. The user is given the option to activate/deactivate this function, in case the user does not want the Java applications to be automatically launched in some instances. The auto launching function is useful for applications that update information at certain intervals, such as stock quotes and weather information.

(2) Network Connection Setting

If the "UseNetwork" key in the ADF is specified, the Java application can communicate with the server from which that application was downloaded. The user is given the option to enable/disable the network connection, as the application might autonomously establish communication (i.e., communicate without any user operation), unlike conventional browsers.

(3) Version Upgrading

If the user selects the version upgrading function of the application, JAM will reacquire the application's ADF, compare the LastModified in the saved ADF (Table 2) with the LastModified of the newly acquired ADF, and download a JAR file if there are any upgrades available. The content of the existing scratch pad can be used in the upgraded application.



ADF: Application Descriptor File
IP: Information Provider
JAR: Java Archive

Figure 3 Download Sequence

Table 2 ADF Information

Key Value	Remarks
AppName	Name of application
AppVer	Additional information
PackageURL	Location of JAR file
AppSize	Size of JAR file
KvmVer	KVM version that can run application
SPsize	Size of scratch pad used
AppClass	Main class at the time of running application
AppParam	Parameter at the time of running application
LastModified	Date of last updating of application
UseNetwork	Declaration to use network function
TargetDevice	Indicates that application is designed for specific models
LaunchAt	Declaration of auto launching

ADF: Application Descriptor File
APP: Application
KVM: K Virtual Machine
SP: Scratch Pad

5.3 Security Check

When a Java application attempts to communicate with a server, JAM checks whether the server is able to communicate.

In the Java runtime environment of the 503i series, the server must satisfy the following conditions to communicate for security purposes.

(1) The host, the port and the scheme (protocol) must be the same as the site from which the application was downloaded.

(2) Redirection is not performed as a result of the communication process.

6. Security in Java Runtime Environment

Along with the installation of the Java runtime environment in the mobile phones, much consideration has been given about security issues.

6.1 Security at the VM Level

J2ME/CLDC verifies the byte codes in advance, in the course of developing the application. In the prior verification stage, the stack map (which is normally generated in the memory when the application is executed) is recorded in the class file, along with other information. The class format leaves the Java Development Kit (JDK) and other standard Java formats as they are, apart from adding the information to the attribute section. J2ME/CLDC also verifies the byte codes prior to the execution of the application: it verifies the adequacy of stack processing, by referring to the information in the class file verified prior to compilation. The verification ensures that CLDC is on par with the system security function provided by JDK.

GLOSSARY

ADF: Application Descriptor File
API: Application Programming Interface
APP: Application
CLDC: Connected Limited Device Cofigu-ration
CPU: Central Processing Unit
HTML: Hyper Text Markup Language
HTTP: Hyper Text Transfer Protocol
HTTPS: Hyper Text Transfer Protocol Se-curity
IP: Information Provider
J2ME: Java 2 Platform Micro Edition
J2SE: Java 2 Standard Edition
JAM: Java Application Manager
JAR: Java Archive
JCP: Java Community Process
JDK: Java Development Kit
KVM: K Virtual Machine
MIDP: Mobile Information Device Profile for the J2ME Platform
OS: Operating System
PDA: Personal Digital Assistant
SP: Scratch Pad
VM: Virtual Machine

6.2 Security at the Application Level

CLDC does not include any specification of functions that may deliberately be exploited by malicious applications, such as those designed to destroy systems. The lack of API makes the development of malicious and dangerous applications impossible. By the same token, i-mode extended API does not consist of any API that may be exploited by dangerous applications.

As JAM controls access to system resources, it is impossible to access the memory dial and mailing functions in the mobile phone. JAM is a component independent of KVM, and it cannot be controlled by Java applications. Furthermore, no more than one Java application can be run at once, which prevents applications from interfering with each other, and disables applications from sharing data. In other words, the data used by a particular application cannot be used by any other application. As native component JAM manages the KVM and the applications, the security of the mobile phone can be maintained even in the face of a malicious Java application.

7. Conclusion

The introduction of Java into mobile phones marks the first step towards exploring a new service frontier. In a nutshell, it is an unprecedented challenge, to make mobile phones run programs that have been downloaded from outside. We intend to further enhance multimedia features in mobile phones in the future.

REFERENCES

[1] Website of Sun Microsystems (http://java.sun.com/)
[2] "i-mode Java spec" in DoCoMo's website (http://www.nttdocomo.co.jp/i/java.html)