Connected Device Configuration (CDC) and the Foundation Profile

Technical White Paper

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Please
Recycle

Contents

CHAPTER 1

The Java™ 2 Platform, Micro Edition (J2ME™) for Consumer Electronic Embedded Devices

Java™ technology has become the platform of choice for getting work done in the network. Sun Microsystem's Java 2 Platform, Micro Edition (J2ME™) technology is designed to address the needs of consumer electronic and embedded devices, such as two-way pagers and personal digital assistants (PDAs). These devices are special-purpose, limited-function devices, not general-purpose computing machines, and are characterized by 32-bit processors and network connectivity. Java technology allows enhanced interactivity with these devices for consumers, developers, and manufacturers alike.

J2ME Technology Enhances Interactivity on Consumer Devices

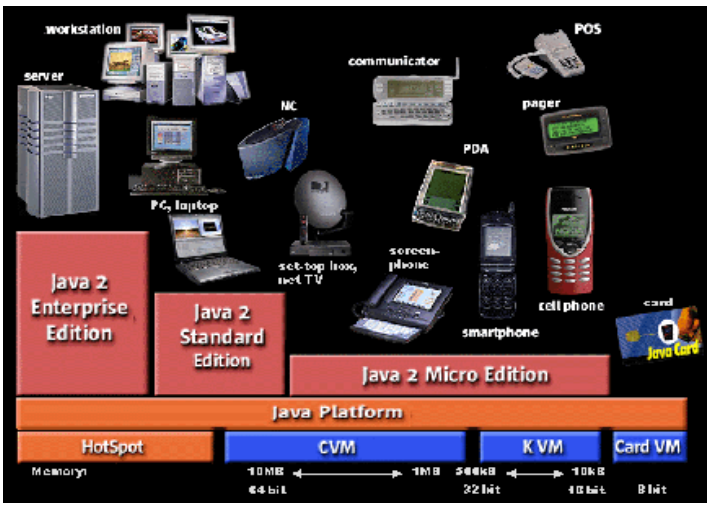
The J2ME architecture is designed to be modular, using two fundamental elements: configuration and profile. Together, they deliver a specification for consumer electronics and embedded device manufacturers to implement on their products.

- A configuration is a virtual machine and minimal set of basic class libraries and APIs. It specifies a generalized runtime environment for consumer electronic and embedded devices, and acts as the Java platform on the device.
- A profile is an industry-defined specification of the Java APIs used by manufacturers and developers to address specific types of device.

Two configurations make up J2ME technology: the Connected, Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). CLDC was developed using a modular approach for devices with a small amount of memory. It is used in small, resource-constrained devices, each with a memory budget in the range of 160 Kbytes to 512 Kbytes. This provides the minimal functionality needed to minimize the memory footprint for memory-constrained devices, which usually contain 16- or 32-bit processors and a minimum total memory footprint of 128 Kbytes. The Mobile Information Device Profile (MIDP) is used with CLDC.

The CLDC is composed of the K virtual machine (KVM) and basic class libraries that can be used on a variety of devices. The KVM is a highly portable virtual machine designed for small memory, limited-resource, network-connected devices.

In contrast, the subject of this paper, CDC, was developed for devices with relatively larger amounts of memory, typically two megabytes or more of total memory available for the Java platform. It employs a modular configuration and profile architecture. Currently, CDC has one profile, the Foundation Profile. In the future, more profiles may be added to provide a user interface, increased Internet connectivity, and Web fidelity.



Java Technology Targets a Broad Range of Devices

CHAPTER 2

Connected Device Configuration (CDC) Overview

The connected device configuration (CDC) contains the full Java 2 Platform virtual machine, the CVM virtual machine, and the minimal class libraries and APIs to get the system up and running. For an application to do useful work, a profile is also required. The Foundation Profile, which includes all the remaining class libraries and APIs that are designed for consumer devices, must be used with the CDC. In addition, other profiles would be required if an application needs a graphical user interface (GUI) or enhanced Web fidelity. Devices that currently use the PersonalJava™ application environment can easily be ported to CDC.

Devices that currently use CLDC (and therefore have a profile) can use that profile with CDC without modification. The CDC is a superset of CLDC and, therefore, a CLDC-compliant profile is upwardly compatible with CDC.

CDC Standardization

CDC is a Java Community Process℠ effort (JSR-36) that has standardized a portable, Java technology building block for consumer electronic and embedded devices. It provides a virtual machine and set of basic libraries appropriate for use with an industry-defined profile, such as the Foundation Profile. The CDC standardization effort was the result of a collaboration of companies representing the consumer electronic industry.

Memory Footprint

Target devices for CDC typically have two megabytes or more of total memory available for the Java platform, including both RAM and flash or ROM. These devices typically require the full feature/functionality of the Java virtual machine.

Connectivity

Target devices for CDC typically contain connectivity to a network, often a wireless, intermittent connection, with limited bandwidth (often 9600 bps or less). Some devices that might be supported by CDC include residential gateways, emerging next-generation smart phones and communicators, two-way pagers, PDAs, organizers, home appliances, point-of-sale terminals, and automobile navigation systems.

CDC Class Library

The CDC class library is the minimal set of APIs needed to support a virtual machine (VM). In general, CDC contains the minimal Java packages from Java 2 Platform, Standard Edition (J2SE™) to build and run a VM.

The basic Java packages for CDC are:

- java.lang — VM system classes
- java.util — Underlying Java utilities
- java.net — UDP Datagram and File I/O
- java.io — Java File I/O
- java.text — Bare minimal support for I18n (ex. error messages)
- java.security — Minimal fine-grain security and encryption for object serialization

Deprecated API Removal

CDC and Foundation Profile make corrections to the J2SE technology-based API library by removing all noncritical, deprecated APIs. For the full listing of as follows deprecated APIs, see the “Deprecate” API section of the javadoc™ specification. The J2SE technology-based deprecated APIs were removed because:

- A clean start with the correct set of APIs was needed. A deprecated method from the J2SE means, “This API is going away, don’t use it anymore. Use its equivalent instead.” So a correct set of APIs should start out with no deprecated APIs in it.
- Deprecated APIs do not have to be supported and maintained. No maintenance resources are needed for these removed, deprecated APIs. No bug fix or syncing with the J2SE platform is needed.
- Each deprecated API has an equivalent new API. In all cases, the equivalent API is cleaner and more correct.

- When there was a problem with the equivalent API (such as `java.util.Properties.store()` not being in the PersonalJava application environment, version 3.0), that specific deprecated API was put back in, because an equivalent is not readily available.
- CDC does not claim backward binary compatibility. By starting fresh with a source code release, developers on CDC must update their code not to use APIs that have been marked as deprecated, but instead use the new equivalents. This is especially important for the I18n-related deprecated APIs and all other deprecated APIs that the J2SE platform has decided are no longer correct.

CHAPTER 3

CVM Virtual Machine Overview

Compact, consumer-device oriented, and works in a connected environment, the CVM virtual machine is designed for consumer and embedded devices. It is a small footprint, virtual machine incorporating the latest in VM technology, while remaining suitable for the embedded world. It is very portable, RTOS-aware, deterministic, and space-efficient. It allows devices to map Java threads directly to native threads, and can run Java classes out of ROM. It combines these “embedded-friendly” features with advanced VM features such as a precise memory system, generational garbage collector, and fast Java synchronization. Finally, CVM supports all Java 2 Platform, version 1.3, VM features and libraries for security, weak references, JNI, RMI, and JVMDI.

Memory System

CVM incorporates an advanced memory system featuring:

- Exactness
- Small average garbage collection pause times
- Full separation of VM from the memory system
- Pluggable garbage collectors
- Generational garbage collection

Portability

CVM is implemented in C, with very little Assembler. In addition, CVM contains a rich, well-documented porting layer that is RTOS-aware, supports multiple porting options for areas that are difficult to implement by a porter, and in general tries to leave only the bare minimum of work to the porter for faster retargeting.

Fast Synchronization

CVM achieves extremely fast common-case synchronization, making use of the observation that the vast majority of locking operations in Java language programs are uncontended. Therefore, CVM performs most synchronization operations with just a few machine instructions, and without consuming additional lock resources from the underlying operating system (OS).

“ROMable” Classes

CVM can run with preloaded, mostly read-only classes, alongside dynamically loaded classes. This provides better startup time, less fragmentation, more data sharing, and the ability to execute bytecodes out of ROM.

Native Thread Support

CVM supports native threads and arbitrary thread preemption at machine instruction boundaries. It has proper internal synchronization and handles exact garbage collection (GC) and synchronization correctly in the presence of preemptive native threading.

Small Class Footprint

The memory footprint required by CVM for classes is quite small, both for dynamically loaded and “ROMized” classes. The reduction in memory footprint for classes is approximately 40 percent compared to JDK™ classic, and around 17 percent compared to the PersonalJava application environment.

Java 2 Platform, Version 1.3 VM Support

CVM supports the full Java 2 Platform, version 1.3 VM specification and libraries, including support for weak references, reflection, serialization, JNI, RMI, and JVMDI. In addition, fine-grained Java 2 Platform security APIs and the Java Platform Debugger Architecture are supported.

Stack Usage

CVM features deterministic and reduced-footprint native stack usage, achieved by careful static analysis of the VM code. The static analysis has resulted in the placement of native stack checks at possible recursion points in the VM code to detect and prevent native stack overflows.

Startup and Shutdown

CVM can cleanly shutdown and restart in a single address space OS, such as an RTOS, without help from a process model, freeing up all allocated memory and stopping all threads without any resource leaks.

Interfaces

CVM provides extensible and well-defined interfaces. The interfaces between components, such as garbage collection, type system, locking, and interpreter are clearly defined, well-separated, and well-documented. It is much easier to add new features to CVM compared to the previous PersonalJava virtual machine.

CHAPTER 4

Foundation Profile Overview

The Foundation Profile is a set of Java APIs that, together with the CDC, provides a complete, J2ME application runtime environment targeted at consumer electronics and embedded devices. The CVM in the CDC is the engine for the Foundation Profile libraries. Typical implementations use a subset of the J2SE platform, depending on any additional profiles supported. The Foundation Profile specification has been developed through the Java Community Process (JCP) program by an expert group composed of companies representing the consumer electronics industry, and has been approved by them.

Overview of Foundation Profile Class Library

In general, the Foundation Profile contains the complete basic Java packages from J2SE technology, except that the GUI dependencies on `java.awt` are removed.

The basic Java packages from CDC are modified in the Foundation Profile by:

- `java.lang` — Rounds out full `java.lang.*` J2SE package support for the Java language (Compiler, UnknownError).
- `java.util` — Adds full zip support and other J2SE utilities (Timer).
- `java.net` — Adds TCP/IP Socket and HTTP connections.
- `java.io` — Rounds out full `java.io.*` J2SE package support for Java language input/output (Readers and Writers).
- `java.text` — Rounds out full `java.text.*` J2SE package support for I18n (Annotation, Collator, Iterator).
- `java.security` — Adds Code Signing and Certificates.

Relationship to J2SE Technology

The Foundation Profile set of libraries is based on the J2SE v1.3 software. The following classes in the Foundation Profile differ from the same-name classes, as described below.

java.lang.ClassLoader

The following line was removed: `import java.io.StringWriter;`

There is no change in behavior, because ClassLoader does not use the imported class.

java.lang.SecurityManager

The `java.awt` dependencies were removed, as the Foundation Profile does not include the `java.awt` package. Hard-coded references to `java.awt` were changed to use reflection to determine if the `java.awt` package is present. (It could be present in another profile.)

If java.awt is present, there is no difference in Security Manager behavior compared to J2SE.

If `java.awt` is not present:

- `checkTopLevelWindow(Object window)` returns `false`.
- `checkSystemClipboardAccess()` throws `SecurityException("AWT not available")`.
- `checkAwtEventQueueAccess()` throws `SecurityException("AWT not available")`.

java.text.resources.LocaleElements

The `java.awt` dependency has been removed, as the Foundation Profile does not include the `java.awt` package. Specifically, the following line is commented out:
`import java.awt.ComponentOrientation;`

The following line is also commented out:
`{ "Orientation", ComponentOrientation.LEFT_TO_RIGHT },`

If `java.awt` is present through an additional profile, `LocaleElements` will not have access to the `ComponentOrientation` resource.

java.text.resources.LocaleElements_en_US

The `java.awt` dependency has been removed, as the Foundation Profile does not include the `java.awt` package. Specifically, the following line is commented out:
`import java.awt.ComponentOrientation;`

There is no change in behavior, as `LocaleElements_en_US` does not use the imported class.

sun.net.www.ParseUtil

The following line was removed: `import java.util.BitSet;`

There is no change in behavior, because `ParseUtil` does not use the imported class.

CHAPTER 5

Resources

Documentation API specifications for these releases can be downloaded from the Sun Web site at java.sun.com/products. They are also available via the Sun Community Source License mechanism at www.sun.com/software/communitysource.

- CDC release — java.sun.com/products/cdc
- CVM — java.sun.com/products/cdc/cvm
- Foundation Profile release — java.sun.com/products/foundation

After the downloaded file is unzipped, use a Web browser to view the start page in `install/javadoc/index.html` under the directory in which the downloaded file was unzipped.

Official CDC and Foundation Profile Specifications

The official CDC and Foundation Profile specifications are located at the Java Community Process Web site for Java Specification Requests (JSRs) #36 and #46. These represent the only official versions of the CDC and Foundation Profile specifications. Specifications generated from the source files of the reference implementations are not valid.

- Official CDC specification, JSR #36 — java.sun.com/aboutJava/communityprocess/jsr/jsr_036_j2mecd.html
- Official Foundation Profile specification, JSR #46 — java.sun.com/aboutJava/communityprocess/jsr/jsr_046_j2mefnd.html

The following documents may also be of interest:

- Java 2 Platform, Micro Edition (J2ME) — java.sun.com/products/j2me
- CLDC and the K Virtual Machine (KVM) — java.sun.com/products/cldc
- Java Community Process — java.sun.com/aboutJava/communityprocess

- Bracha, Gilad, James Gosling, Bill Joy, Guy Steele. *The Java Language Specification, Second Edition (The Java Series)*. Addison-Wesley Pub Co, 2000. java.sun.com/docs/books/jls/index.html
- Lindholm, Tim, and Frank Yellin. *The Java Virtual Machine Specification*. Addison-Wesley Pub Co, 1999. java.sun.com/docs/books/vmspec/index.html
- Portable Operating Systems Interface (POSIX) catalog — standards.ieee.org/catalog/posix.html
- Plauger, P. J. *The Standard C Library.* New Jersey: Prentice Hall, 1992. www.prenhall.com/books/ptr_0131315099.html

Please send comments to: j2me-cdc-comments@sun.com.

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

1 (800) 786.7638
1.512.434.1511

www.sun.com