J2ME CLDC Reference Implementation

Release Notes / CLDC 1.0

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

CLDC 1.0
May 19, 2000

Contents

1 Introduction

These release notes provide information about Sun's reference implementation of the Connected, Limited Device Configuration (CLDC).

CLDC is the result of a Java Community Process effort (JSR-30) whose goal is to define a standard, portable Java™ platform for small, resource-constrained, connected devices. The CLDC specification effort has been done in collaboration with 18 companies representing different industries. Target devices for CLDC are characterized generally as follows:

- 160 to 512 kilobytes of total memory, including both RAM and flash or ROM, available for the Java platform.
- Limited power, often battery powered operation.
- Connectivity to some kind of network, often with a wireless, intermittent connection and with limited (often 9600 bps or less) bandwidth.
- User interfaces with varying degrees of sophistication down to and including none.

Cell phones, two-way pagers, personal digital assistants (PDAs), organizers, home appliances, and point of sale terminals are some, but not all, of the devices that might be supported by CLDC.

The CLDC reference implementation runs on Sun's K Virtual Machine (KVM) implementation that is provided as part of this release.

Note that CLDC is intended to serve as the lowest common denominator for various kinds of resource-constrained, Java-powered devices. As such, CLDC is not a complete, self-sufficient solution, but it needs to be complemented by *profiles*. For instance, all the user interface aspects are outside the scope of CLDC Specification. A parallel Java Community Process effort (JSR-37) called *Mobile Information Device Profile* (MIDP) is currently underway to define the necessary remaining Java platform features and libraries for a specific vertical market/device category. Other profiles for other vertical markets or device categories may be defined later.

Supported Platforms

The *J2ME CLDC Reference Implementation* runs on Windows 98/NT (Win32) and Solaris platforms.

A CLDC-compatible port for the Palm Connected Organizer is available separately. The Palm release package can be installed on top of this release package. The Palm port is not considered to be a reference implementation of the CLDC.

Items Included in the Release

This release includes the source code and binaries for:

- K Virtual Machine (KVM)
- Preverifier tool
- JavaCodeCompact tool (for class prelinking/preloading)
- JAM (Java Application Manager) reference implementation
- CLDC Java libraries (API)
- Additional Java libraries that are not part of CLDC (GUI and database classes in package `com.sun.kjava`; storage and network protocol implementations in package `com.sun.cldc.io`)
- Sample applications provided for testing purposes

Please refer to *KVM Porting Guide*, Sun Microsystems, Inc. for more information on the preverifier tool, JavaCodeCompact tool and the JAM.

The release includes the following documentation:

- *J2ME CLDC Reference Implementation Release Notes* (this document)
- Connected Limited Device Configuration Specification
- CLDC API document
- *KVM Porting Guide*, Sun Microsystems, Inc.
- `com.sun.kjava` GUI API documentation (not part of CLDC)

Note – The classes provided in packages `com.sun.kjava` and `com.sun.cldc` are not officially part of the CLDC reference implementation. These classes have been provided to facilitate porting and testing efforts, and may change or may be removed in future releases of this software.

Prerequisites and Dependencies

For more details on the Connected, Limited Device Configuration, please refer to the *Connected, Limited Device Configuration Specification*, Sun Microsystems, Inc.

Information about Java 2 Micro Edition (J2ME) configurations and profiles is provided in *Configurations and Profiles Architecture Specification, Java™ 2 Platform Micro Edition (J2ME)*, Sun Microsystems, Inc.

Please refer to the *KVM Porting Guide*, Sun Microsystems, Inc. for information regarding porting the K Virtual Machine to new platforms.

2

Installation Notes

Unzipping the Distribution

Unzip the distribution into any directory of your choice. It creates the directory `j2me_cldc` with the following subdirectories:

- `api`
- `bin`
- `build`
- `docs`
- `jam`
- `kvm`
- `samples`
- `tools`

Please refer to the *KVM Porting Guide*, Sun Microsystems, Inc. for further information on the contents of these directories.

Building the Source Release

The K Virtual Machine and the associated preverification tool have been written in C. This software has been compiled successfully with the following compilers:

- Sun DevPro C compiler 4.2 on Solaris
- GNU C compiler on Solaris
- Microsoft Visual C++ 6.0 (Professional Version) on Windows 98 and NT 4.0
- Metrowerks CodeWarrior Release 6 for Palm (CodeWarrior IDE version 4.0.1 build 0436)

In order to compile the Java library files, sample applications, and some additional tools provided in the source release, Java Development Kit (JDK) 1.2.2 or later is needed.

You should be able to build all the binaries included in this release from the source code files shipped with the release. The necessary gnu tools for building the binaries are not provided with this release, but can be downloaded from

http://www.gnu.org/software/software.html

or

http://sourceware.cygnus.com/cygwin/

Note – Windows NT 4.0 SP 5 was used to build the Windows binary of KVM for this release.

Building the Release on Win32

Enter the build/win32 subdirectory and type `gnumake`. This builds all the files shipped with the Win32 binary release. Refer to the README file in the build/win32 directory for the environment you need to setup to build the KVM release on Win32.

Building the Release on Solaris

Enter the build/solaris subdirectory and type `gnumake`.

Building the Release on Palm

A Palm release package is available separately, and can be installed on top of this release. For building the CLDC/KVM implementation for the Palm, a Metrowerks Codewarrior project file is provided.

Running Sample Applications

The directory `samples` contains class files for several demonstration programs. Some of these class files (described below) contain a "main" method, and are therefore directly runnable in KVM. The other classes are utility classes referenced by the main classes.

The Palm-like GUI

The Connected, Limited Device Configuration itself does not define any GUI (graphical user interface) functionality. For testing purposes, this release includes a number of GUI classes that were originally written for the Palm implementation of KVM. These classes can be found in directory `api/classes/com/sun/kjava/`.

Note – The GUI classes provided in package `com.sun.kjava` are *NOT* part of the Connected Limited Device Configuration (CLDC). Official GUI classes for Java 2 Micro Edition will be defined separately through the Java Community Process and included in *J2ME profiles*.

Running the demo apps on Windows/Solaris

The sample applications in directory `samples` utilize the `com.sun.kjava` GUI classes that were originally written for the Palm version of KVM. The sample applications can be run on the Unix and Windows versions of KVM, because these versions contain the necessary native methods to provide a simulated "Palm-like" GUI environment. This GUI has the following features:

- Monochrome (black/white) input/output screen of 160 x 160 pixels with mouse support. When the mouse cursor is within the screen area, it behaves as the stylus (pen) on a Palm device. Clicking the left mouse button corresponds to tapping the screen with the pen. Moving the mouse while holding down the left mouse button corresponds to dragging the pen across the screen.
- Dummy silk-screened area. This is the large rectangle directly below the screen, which contains a smaller rectangle and two circles on either side. This area is NOT supported in the Unix and Windows versions of KVM. It corresponding to the silk-screened area of a Palm device, containing the Graffiti are and the Home/Applications, Menu, Calculator, and Find buttons.

- Simulated "hard" buttons. Clicking the mouse on one of these rectangles or circles corresponds to pressing a "hard" button on a Palm device. These buttons ARE supported in the Unix and Windows versions of KVM. Left to right, they are: Power button, Calendar button. Address Book button, (upper) Scroll Up button, (lower) Scroll Down button, To Do List button, and Memo Pad button.

  Note: Most of the sample programs respond only to the Scrolling buttons. The Missiles game responds to all the "hard" buttons, but they are used to control the game (not to invoke the applications listed above).

  Note: Unlike in a real Palm device, holding down a hard button does NOT send repeated button events. You must repeatedly tap the button to produce multiple events.

- Simulated Graffiti input. Typing a key on the keyboard corresponds to entering that character into the Graffiti area on a Palm device.

  Note: Many of the sample programs do not respond to character input.

For convenience, two script files are provided in the samples directory for running the sample programs on Unix and Windows. "ku" is a Unix shell script, and "kw.bat" is a Windows batch file. Depending on the platform, the sample programs can be executed by typing either:

```
ku <classname>
```

or

```
kw <classname>
```

When the program quits, the KVM stops and the GUI screen disappears.

The following section lists each of the nine sample programs using an example of the command to start the program. A few notes on the use of each program are also provided.

"ku Dragon"

Tap anywhere on the screen to draw a dragon fractal at that point on the screen. Use the "+" and "-" buttons to increase/decrease the number of iterations or segment size of the next dragon to be drawn. Use the "?" to select a random number (within the allowable range for that control).

"kw ManyBalls"

The program begins with a single ball bouncing around the screen (driven by its own Java thread within KVM). Press the "+" and "-" to add or decrease the number of ball threads.

"ku Pong" and "ku StarCruiser"

Use the scroll bar slider, the scroll arrows, or the "hard" Scroll buttons to view all the game instructions. Press "Done" to go to the game screen.

"kw Scribble"

Read the game instructions and press "Done" to go to the game screen. This program responds to character input by displaying the character on the screen. Use the "hard" Scroll buttons to change the size of the ball. You can also "drag" the ball to another place on the screen.

"kw ThreeDLogo"

Drag the pen to cause the 3D figure to rotate. Tap anywhere on the screen to return the figure to its starting position. The "Beam" button is NOT supported.

"ku UITest"

A simple test program for some of the UI classes.

"kw dots.DotGame"

Tap the "Help" button for instructions. Tap one of the four class names to select that algorithm for either the Host or the Guest player. Use PenTaps for a human player. Use one of the other 3 classes to play against that program. Either or both of Host and Guest can be human or program. Tap one of the "moves first" buttons to start the game. For the most compute-intensive game, play Average against Average on a grid of 11 rows x 13 columns.

"ku missiles.Missiles"

Tap the "Help" button for instructions. You don't need to gather any classes because they are all provided (in samples/missiles/*). Sound is not supported on the Unix and Windows versions. The Unix version displays the bitmaps somewhat incorrectly (green icons in a white box) but the game is still playable.

Running the demo apps on Palm

If you have installed the additional Palm release package on top of this release, you can run the sample applications on a real Palm device. Demo applications are provided as ‘prc’ (Palm executable) files. Once you have installed the ‘KVM.prc’ and ‘KVMutil.prc’ executables on your Palm device, simply install these application files on your Palm device, and launch them from the Palm application launcher.

Building Sample Applications from Source Code

Application development for CLDC usually takes place on a desktop computer. At the high level, the procedure for building Java applications for CLDC is as follows:

1. compile the Java application using a Java compiler (not provided as part of this release.)
2. preverify the Java classfiles with the `preverify` tool provided in this release.
3. use the `jar` tool to create a JAR file that contains all the Java classes belonging to your application (the `jar` tool is not provided as part of this release.)
4. use the `MakePalmApp` tool to convert the Java classfiles or a JAR file into a ‘prc’ (Palm executable) file (if you intend to run your application on the Palm platform.)

Sample command line operations:

- Compilation:

```
javac -g:none -d tmp -classpath tmp:../lib/classes
-bootclasspath ../lib/classes src/Pong.java src/PongBall.java
```

- Preverification:

```
../bin/preverify -d classes -classpath ../lib/classes tmp
```

- JAR creation:

```
jar cvf Pong.jar Pong.class PongBall.class ...more classes...
```

- Building a Palm executable:

```
java -classpath ../lib/classes palm.database.MakePalmApp -v
-version "1.0" -icon icons/pong.bmp -bootclasspath
../lib/classes -classpath classes Pong
```

Refer to `docs/tools.html` for further instructions on how to build the sample applications.

3

Quality Assurance

Testing

JCK compatibility tests version 1.3, Tonga regression and stress tests and CLDC 1.0 TCK compatibility tests have been run on a regular basis on emulators and on the following platforms:

- Solaris
- Microsoft Windows 98
- Microsoft Windows NT
- Palm IIIx, V, VII (Palm implementation of KVM)

The *J2ME CLDC Reference Implementation* passes all the 4357 test cases included in CLDC TCK 1.0. The TCK compatibility toolkit performs comprehensive regression testing of various Java language, virtual machine and library features supported by CLDC.

Components that are outside the scope of CLDC (e.g., packages `com.sun.kjava` and `com.sun.cldc.io)` have not undergone similar regression tests. Various demo programs have been used for testing those components.

Known Bugs

A number of bugs have been dispatched for re-engineering but remain open at the time of this release. For a definitive reference on open bugs and feature requests, log in to the Java Developer Connection (JDC) web site: `http://developer.java.sun.com/developer/`.

Bugs related to the K Virtual Machine and CLDC can be found in: `http://developer.java.sun.com/developer/bugParade/index.jshtml`, under the bug category “K Virtual Machine”.