Contents

1. System Requirements 1
   Introduction 1
   SDK Requirements 2
       SDK Host System Hardware Requirements 2
       Linux System Software Requirements 2
       Windows System Software Requirements 3
       Development Environment 3
   Target Device Requirements 4

2. Installation Procedures 5
   Installing the SDK 5
       ▼ Install the Linux/X86 SDK 5
       ▼ Install the Windows SDK 6
   SDK Installation Structure 9
   Setting Up the Linux/ARM Environment 9
       ▼ Install the Linux/ARM Package on the Host 9
       ▼ Test an Oracle Java ME Embedded Client Application on the Linux/ARM Target 11

3. Working With NetBeans 13
   Oracle Java ME Embedded Client SDK and NetBeans Projects 13
4. **Working With Eclipse** 25
   Oracle Java ME Embedded Client SDK and Eclipse Projects 25
   ▼ Adding a Java Platform 25
   ▼ Writing Your First Application 28

5. **Command Line Options** 31
   Changing Default Resolution for Linux SDK 31
   Stop a Running VM 32
System Requirements

The Oracle® Java Micro Edition (Java ME) Embedded Client provides a Java ME – Connected Device Configuration (Java ME-CDC) virtual machine along with Foundation Profile (FP). The Personal Basis Profile (PBP) package provides basic GUI functionality.

The SDK also includes:
- RMI Optional Package
- J2EE Optional package for CDC/Foundation Profile
- J2ME Web services
- Security Optional Package

The SDK can be installed on X86 PCs running Linux or Windows operating systems, providing a Java ME - CDC application development and emulation environment. For device testing, applications can be exported to a Linux/ARM target running the headless version of Oracle Java ME Embedded Client.

Introduction

The Oracle Java ME Embedded Client includes the following software:
- A software development kit (SDK) for installation on an X86 PC host running Windows XP or Linux. This environment is used for application development and emulation. See “Install the Linux/X86 SDK” on page 5 or “Install the Windows SDK” on page 6.
- An embedded client binary for use on a target platform. See “Setting Up the Target Runtime Environment” on page 10.

This document describes the installation process for the host and target including a brief description on how to develop a simple application using the NetBeans or Eclipse IDE that you can use to test the installation on the target device.
SDK Requirements

The Oracle Java ME Embedded Client SDK environment must meet the following hardware and software requirements.

- “SDK Host System Hardware Requirements” on page 2
- “Linux System Software Requirements” on page 2
- “Windows System Software Requirements” on page 3
- “Development Environment” on page 3

SDK Host System Hardware Requirements

The system hosting the SDK must meet these minimum requirements:

- 1.0 Ghz X86 family processor
- 512 Mb Ram
- Full PBP functionality requires an accelerated graphics card. Oracle has tested the following cards:
  - Nvidia cards
  - Intel G33/G31 series
  - ATI Radeon HD 4300
  - Mobile Intel 4 series express
- 140 Mb of free disk space for the Oracle Java Micro Edition Embedded Client SDK.

Linux System Software Requirements

The Linux software environment is as follows:

- Ubuntu 10.04 32-bit native Linux installation with Frame Buffer support. To verify Frame Buffer support type: sudo fbset -i
- libdirectfb-extra
  - libdirectfb-extra is not included in the standard Ubuntu distribution. Use the Ubuntu software center to install it or the apt-get utility.
Chapter 1 System Requirements

After installing `libdirectfb-extra`, ensure that `DFBARGS=system=x11` is set. This environment variable can be set from the command line (for example, see “Changing Default Resolution for Linux SDK” on page 31), but to make it more persistent, you can set it in either of two other places:

- in your `.bashrc` file. For example, add this line:
  ```
  export DFBARGS="system=x11"
  ```
- Create or edit a `.directfbrc` file in your home directory. Add this line:
  ```
  system=x11
  ```

Windows System Software Requirements

The Oracle Java ME Embedded Client SDK runs on Windows XP. Ensure that the following Microsoft packages are installed.

- Microsoft Visual C++ 2008 SP1 Redistributable Package (x86)
  ```
  ```
- DirectX End-User Runtimes (March 2009)
  ```
  ```
  To install this package, extract the files. In the extraction directory, locate and run `DXSETUP.exe` to complete the installation process.
  - To support full 3D acceleration, run the DxDiag utility. On the Display tab, ensure that Direct 3D Acceleration is enabled.
- Microsoft .NET Framework version 2.0 or higher. The x86 version requires 850 MB free space while the x64 version requires 2 GB of disk space. If .NET is not installed, the installer will prompt you to add it.

Development Environment

The development environment requires Java. An IDE for profiling and debugging is optional.

- JDK 1.6 or higher. Please download the JDK from the Oracle website. OpenJDK has not been tested, and therefore is not supported.
  ```
  ```
- NetBeans 6.9.1 (including Java ME support) with the latest updates, or Eclipse Classic 3.6.1 (Helios).
Note – The newer version 7 of NetBeans is not recommended for using Oracle Java ME Embedded Client. If necessary you can obtain the recommended version 6.9.1 at [http://netbeans.org/downloads/6.9.1](http://netbeans.org/downloads/6.9.1). Be sure to download the “All” edition, which includes support for Java ME.

If you have an older 6.9.1 installation it must be updated to work with Oracle Java ME Embedded Client. In NetBeans, select Help > check for Updates. Repeat the update process until NetBeans reports “Your IDE is up to date.”

Target Device Requirements

Oracle Java ME Embedded Client runtime binaries are available for the following target platforms (all running the Linux operating system):

- ARMv5
- ARMv7
- MIPS
- PowerPC

The Oracle Java ME Embedded Client has been tested on target devices that meet the following criteria:

**Oracle Java ME Embedded Client 1.0 on ARMv5**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>ARM v5</td>
</tr>
<tr>
<td>OS</td>
<td>Linux kernel version 2.6.22 or higher. GLIBC 2.9 or higher.</td>
</tr>
<tr>
<td>Floating Point</td>
<td>Soft Float</td>
</tr>
<tr>
<td>Headful</td>
<td>No</td>
</tr>
<tr>
<td>RAM</td>
<td>4 - 16 MB</td>
</tr>
<tr>
<td>ROM/Flash/Disk</td>
<td>5 MB or more for Oracle Java ME Embedded Client</td>
</tr>
<tr>
<td>Oracle Java ME Embedded Client Version</td>
<td>1.0</td>
</tr>
</tbody>
</table>


### Oracle Java ME Embedded Client 1.0 on ARM v7 Little-Endian

<table>
<thead>
<tr>
<th>CPU</th>
<th>ARM v7</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Linux kernel version 2.6.35 or higher. GLIBC 2.12.1 or higher.</td>
</tr>
<tr>
<td>Floating Point</td>
<td>Hard, VFP</td>
</tr>
<tr>
<td>Headful</td>
<td>No</td>
</tr>
<tr>
<td>RAM</td>
<td>4 - 16 MB (headless only)</td>
</tr>
<tr>
<td>ROM/Flash/Disk</td>
<td>6 MB or more for Oracle Java ME Embedded Client</td>
</tr>
<tr>
<td>Oracle Java ME Embedded Client</td>
<td>Version 1.0*</td>
</tr>
</tbody>
</table>

* Note: this same binary is likely to run on ARMv6 architecture, although Oracle has not tested this.

### Oracle Java ME Embedded Client 1.0 on MIPS

<table>
<thead>
<tr>
<th>CPU</th>
<th>MIPS32 74K</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Linux kernel version 2.6.22 or higher. GLIBC 2.8 or higher.</td>
</tr>
<tr>
<td>Floating Point</td>
<td>Soft Float</td>
</tr>
<tr>
<td>Headful</td>
<td>No</td>
</tr>
<tr>
<td>RAM</td>
<td>4 - 16 MB</td>
</tr>
<tr>
<td>ROM/Flash/Disk</td>
<td>5 MB or more for Oracle Java ME Embedded Client</td>
</tr>
<tr>
<td>Oracle Java ME Embedded Client</td>
<td>Version 1.0</td>
</tr>
</tbody>
</table>

### Oracle Java ME Embedded Client 1.0 on PowerPC

<table>
<thead>
<tr>
<th>CPU</th>
<th>PowerPC 464FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Linux kernel version 2.6.32 or higher. GLIBC 2.10.2 or higher.</td>
</tr>
<tr>
<td>Floating Point</td>
<td>Hard Float</td>
</tr>
<tr>
<td>Headful</td>
<td>No</td>
</tr>
<tr>
<td>RAM</td>
<td>4 - 16 MB</td>
</tr>
<tr>
<td>ROM/Flash/Disk</td>
<td>5 MB or more for Oracle Java ME Embedded Client</td>
</tr>
<tr>
<td>Oracle Java ME Embedded Client</td>
<td>Version 1.0</td>
</tr>
</tbody>
</table>
Installation Procedures

This chapter discusses installation procedures for the Oracle Java ME Embedded Client SDK binary and the environment required to support the emulator.

Installing the SDK

This section describes how to install the SDK on Linux and Windows hosts.

▼ Install the Linux/X86 SDK

This command line procedure installs the SDK in /usr/local.

1. Download the Oracle Java ME Embedded Client bundle to any location on a compliant system.

2. From a Linux terminal use the following command to enable execute permissions on the binary:

   ```
   $ chmod +x ./ojec_1.0_linux_x86_sdk.bin
   $ sudo ./ojec_1.0_linux_x86_sdk.bin
   $ [sudo] password for userid:
   $ Oracle Java Micro Edition Embedded client version 1.0
   Installer is starting, please wait...
   ```

3. Use the following steps to run the installer binary:
4. At the prompt, type 1 to start the installation.

```
1 - Install
2 - Uninstall
3 - Quit
Enter choice: 1
Installing Oracle Java Micro Edition Embedded Client version 1.0...
```

If the installation is successful you will see the message “Installation completed.”

If another version exists in /usr/local, you will be warned:

```
Please note that installation directory
/usr/local/Oracle_JavaME_Embedded_Client/1.0
is not empty and files under it will be overwritten!
Do you want to continue [y/n]?
```

Type y to continue or n to exit.

Oracle Java ME Embedded Client is installed in
/usr/local/Oracle_JavaME_Embedded_Client/1.0.

▼ Install the Windows SDK

This command line procedure installs the SDK in C:\Program Files\Oracle\Oracle JavaME Embedded Client by default.

1. **Download the file ojec_1.0_win32_x86_sdk.zip. Extract the zip file.**
   This exposes ojec-sdk-setup.msi and setup.exe.

2. **Double-click on setup.exe to start the installation.**
Chapter 2 Installation Procedures

Click Next to continue.

3. Select an installation folder.
4. Confirm that you want to start the installation.

5. Click Close at the end of the installation.
SDK Installation Structure

The default installation locations are:

**Linux:** /usr/local/Oracle_JavaME_Embedded_Client/1.0

**Windows:**
- C:\Program Files\Oracle\Oracle JavaME Embedded Client
  (on 32-bit machines) or
- C:\Program Files (x86)\Oracle\Oracle JavaME Embedded Client
  (on 64-bit machines)

The installation directory contains the following subdirectories:

- **docs/** Contains API documentation and product documentation for the JSRs provided with this SDK.
- **emulator-platform/** Contains all binary files, libraries and others necessary to run the Java Virtual Machine.
- **legal/** Contains license agreements for the Oracle Java Micro Edition Embedded Client.

---

Setting Up the Target Runtime Environment

The procedures in this section describe how to install the runtime package on a Linux development host and make it available to a target device.

The procedures in this section describe working with the runtime binary targeted for an ARMv7 device running Linux. The procedures are very similar for the other available runtime binaries, such as ARMv5, MIPS or Power PC.

▼ Install the Linux/ARM Package on the Host

When an embedded platform has limited memory it is advisable to install the stack on the host, then mount the stack from the ARM device. If the ARM device has sufficient storage you can copy the Oracle Java ME Embedded Client platform directory to the device.

1. **Download** ojec_1.0_linux_arm_pack.tar.gz **and save it on a qualified Linux host** (see “SDK Requirements” on page 2).
2. To extract the contents in the current directory, use this command:

   tar xzf ojec_1.0_linux_arm_pack.tar.gz

   The directory Oracle_JavaME_Embedded_Client/1.0 is created in the
   extraction directory. It includes these subdirectories:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binaries</td>
<td>All binaries and libraries for Oracle Java ME Embedded Client for Linux/ARM.</td>
</tr>
<tr>
<td>legal</td>
<td>Copyright and other legal documents.</td>
</tr>
<tr>
<td>docs</td>
<td>All relevant documentation for Oracle Java ME Embedded Client.</td>
</tr>
</tbody>
</table>

3. If you are going to copy the stack to the device, mount the
   Oracle_JavaME_Embedded_Client/1.0 directory.

   If you are going to mount the binaries from the ARM device, export the
   Oracle_JavaME_Embedded_Client/1.0/binaries directory.

   For example, add the path to /etc/exports with proper permissions and then
   restart the NFS server.

Optional JAR Files in the Stack

The Oracle Java ME Embedded Client runtime for Linux/ARM comes bundled with
a number of optional packages, which are found in:

InstallDir/Oracle_JavaME_Embedded_Client/1.0/binaries/lib

If these APIs are not used and you choose to reduce the size of the static footprint,
you can safely delete any of the following files from the system:

<table>
<thead>
<tr>
<th>Optional Package</th>
<th>File Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMI Optional Package Specification Version 1.0</td>
<td>jsr66.jar</td>
</tr>
<tr>
<td>JDBC Optional Package for CDC/Foundation Profile 1.0</td>
<td>jsr169.jar</td>
</tr>
<tr>
<td>J2ME ME Web Services Specification 1.0</td>
<td>jsr172.jar, xmlparser.jar</td>
</tr>
<tr>
<td>Security Optional Package 1.0</td>
<td>jaas.jar, jce.jar,</td>
</tr>
<tr>
<td></td>
<td>jsse-cdc.jar, ext/sunjec_provider.jar,</td>
</tr>
<tr>
<td></td>
<td>sunrsasign.jar</td>
</tr>
</tbody>
</table>
Test an Oracle Java ME Embedded Client Application on the Linux/ARM Target

These instructions assume the target device complies with the “Target Device Requirements” on page 4.

1. From the Linux/ARM target platform, mount the directory exported from the Linux host computer in Step 3 of “Install the Linux/X86 SDK” on page 5.

   For example:

   ```
   mount -t nfs -o nolock
   linuxhost:/usr/local/Oracle_JavaME_Embedded_Client/1.0/binaries
   /mnt/Oracle_JavaME_Embedded_Client
   ```

   Alternatively, you can copy the entire stack from the mount point to the device, or only the parts of the stack you need, as described in Step 3 in “Install the Linux/ARM Package on the Host” on page 10.

2. Use the Oracle Java ME Embedded Client SDK to build and compile an application on the Linux host computer.

   Note – In version 1.0, the only available client is headless. Your application must use a Main file that uses `public static void main (String[] args) { }`. Your headless application must not use an Xlet and it must not rely on a graphics stack.

3. Make your application’s JAR (or class) file accessible from the Linux/ARM target platform (for example, use `mount`).

4. Run the application using either the optimized version or the debug version of the VM:

   ```
   InstallDir/Oracle_JavaME_Embedded_Client/1.0/binaries/bin/cvm
   ```

   For example:

   ```
   cd /mnt/Oracle_JavaME_Embedded_Client/1.0/bin
   ./cvm -cp /projects/Hello/dist/Hello.jar hello.Main
   ```

   Check the output to validate the results.
As mentioned in “Development Environment” on page 3, the recommended NetBeans version is 6.9.1. You can find this version at:


Choose the “Java” download bundle.

This chapter details the steps to configure NetBeans to use the Oracle Java ME Embedded Client SDK as a Java platform, and presents a sample project and to run in your configured environment.

Oracle Java ME Embedded Client SDK and NetBeans Projects

This section details how to add the SDK as a Java Platform in NetBeans and how to create an Oracle Java ME Embedded Client project.

▼ Adding the SDK as a Java Platform

Oracle Java ME Embedded Client provides the Java ME platform for embedded devices, such as TV Set Top boxes and smart electric meters. These devices run a virtual machine based on Java ME CDC. To emulate this environment, the NetBeans IDE must be configured to use the Oracle Java ME Embedded Client platform. Follow these steps to install this platform into your NetBeans IDE.

This procedure was recorded on a Linux machine. If you are a PC user, use the Windows paths discussed in “SDK Installation Structure” on page 9.
1. Start the NetBeans IDE.

2. Choose Tools > Plugins or go to the My NetBeans tab and click Install Plugins.

3. Click the Installed Tab and determine if the Java ME item has a green check mark under the Active column.

   If Java ME is checked, continue to Step 9 or, if Java ME is not enabled, perform the following actions:

   a. Check the box in the Select column as shown in the following screen.
b. Click the Activate button on the bottom left of the Plugins window. The NetBeans IDE Installer window opens.

c. Click the Activate button at the bottom, as shown below.

d. Click Finish when the Java ME plugin activation is complete.
4. If you have an older 6.9.1 installation it must be updated to work with Oracle Java ME Embedded Client. In NetBeans, select Help > check for Updates. Repeat the update process until NetBeans reports “Your IDE is up to date.”

5. After a successful update, NetBeans will prompt you to restart the IDE. Click the option Restart IDE now, and click Finish.

6. Select the menu Tools->Plugins and click the Update tab.
   Click the Java box as shown, and click the Update button. This step enables the plugin for CDC Java Embedded Client Platform Implementation.
7. After a successful update, NetBeans will again prompt you to restart the IDE. Click the option Restart IDE now, and click Finish.

8. Click the Close button to dismiss the Plugins window.

10. Click the Add Platform button below the Platforms pane and the Add Java Platform window opens.
   a. On the Select platform type page, select Java ME CDC Platform Emulator.
   b. Click Next to display the Choose Platform Folder page.
c. On the Choose Platform Folder page, go to the Look In field and choose or navigate to
/usr/local/Oracle_JavaME_Embedded_Client/1.0/emulator-platform.

d. Click Next to display the Platform Name page.
   If no errors are displayed on the Platform Name page, click Finish and the Java Platform Manager opens.

11. Verify that the Oracle Java Micro Edition Embedded Client is added as a CDC platform on the left and click Close.
You are ready to develop applications.

▼ Create and Run a New Project


2. On the Choose Project page, complete the following:
   a. In the categories area, select Java ME.
   b. In the Projects area, select CDC Application.
3. On the Name and Location, perform the following actions:
   a. Enter HelloWorld for the Project Name.
   b. Select an appropriate Project Location.
   c. De-select the Create Main Class option.
   d. Click Next and the Select Platform page is displayed.

4. On the Select Platform page, perform the following actions:
   b. Choose JEC-1.0 from the Profile list.
c. Click the Finish button to create the new HelloWorld project (the name entered in Step 3).

5. Create a Main class in the HelloWorld package by performing the following actions in the Projects view:
   a. Expand the HelloWorld application and the Source Packages tree.
   b. Right-click on <default package>.
   c. Select New > Java Class to open the New Java Class window.
d. Name the class Main, enter a lower case package name such as helloworld, and click Finish.
6. Edit the Main class by inserting a main method to print a string.
   For example:

   ```java
   public class Main {
       public static void main(String args[]) {
           System.out.println("Hello, world!");
       }
   }
   ```

7. In the Projects tab, right-click on the project and select the option Build.

8. Identify helloworld.Main as the HelloWorld project main class.
   a. In the Projects tab, right-click on the HelloWorld project and select
      Properties to open the properties window.
   b. In the Category area, select Running.
   c. In the Main Class field, click Browse to select the Main.java file.
      The Browse Main Classes window opens with helloworld.Main selected.
      The option Run using main(String[] args) method execution should be
      selected.
   d. Click the Select Main Class button and then click OK to close the
      HelloWorld properties window.

9. In the Projects tab, right-click on the project and select the option Run.
   The message, “Hello, world!” prints in the NetBeans Output window. To open the
   output window, select Window > Output > Output.
Project Options

To edit project properties, right click on the project and select the option Properties.

- To change the display resolution, select the Platform category to switch to another emulator platform (if available).
- Open the Build sub-options to set desired values. For example, you can add any JAR file to the build system by selecting Build sub-option Libraries & Resources.
- Modify the Running option to pass Arguments or VM options for the Java runtime.
Working With Eclipse

This chapter details the steps to configure Eclipse to use the Oracle Java ME Embedded Client SDK as a Java platform, and presents a sample project and a sample application to run in your configured environment.

Note – The SDK supports application development and the emulation. Debugging and Profiling are not supported.

Oracle Java ME Embedded Client SDK and Eclipse Projects

This section describes how to add the Oracle Java ME Embedded Client SDK as a Platform in Eclipse and how to create a new project in Eclipse.

▼ Adding a Java Platform

Launch Eclipse and follow these steps:

1. Select Window > Preferences.
   The Preferences Window opens.

2. In the Preferences window, go to the left panel, expand the Java item, and select Installed JREs.
The central panel displays the JREs on this machine.

3. Click the Add button on the right to open the Add JRE window. The window displays the JRE Type options.

4. Select Execution Environment Description and click Next.

The window displays the JRE definition panel.

5. In the Definition File field, click File to open a file chooser window.
   a. Navigate to
      
      /usr/local/Oracle_JavaME_Embedded_Client/1.0/emulator-platform.
b. Select ojec_re_def.ee and click OK to close the file chooser window.

6. Click Finish to load the Oracle Java ME Embedded Client JRE and close the Add JRE window.

The Installed JREs panel lists Oracle Java Micro Edition Embedded Client 1.0 as an installed JRE.

7. Check the box in front of Oracle Java Micro Edition Embedded Client 1.0 and click OK.

Now you are ready to develop applications for the Oracle Java ME Embedded Client.
Writing Your First Application

After completing the steps documented in the previous task, Eclipse has the JRE platform required to build and run Oracle Java ME Embedded Client applications.

1. Select File > New > Project and then Java Project.
   The Create a Java Project window is displayed.

2. In the Create a Java Project window, perform the following actions:

   a. Enter a project name, such as HelloWorld.
   b. In the JRE area, select Use a project specific JRE and choose Oracle Java Micro Edition Embedded Client 1.0.
   c. For this example, choose Create separate folders for sources and class files.
   d. Click the Finish button.

   The new project HelloWorld is created along with separate directories for sources and class files, src and bin respectively.

3. Select File > New > Class.
   The New Java Class window opens.
4. In the New Java Class window, perform the following actions:

![Java Class window with selected options]

- a. The Source folder field should be populated. If it isn’t, browse for the folder.
- b. In the Package field, use lower case characters to enter a name.
- c. In the Name field, enter the class file name.
- d. In the method stubs area, deselect the Inherited abstract methods option and check:
  ```java
  public static void main(String[] args)
  ```

5. Click Finish to create the file **Hello.java** in the **src** folder.
   The new **Hello.java** file in the **src** folder is an empty template and must be modified.
6. In Hello.java, add a print line to the body of the main function:

```java
package hello;
public class Hello {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello World");
    }
}
```

7. Click the green triangle Run button to compile and run your code.
The Hello World message prints in the console.
CHAPTER 5

Command Line Options

This chapter explains how to use certain CVM command line options. The same options can be used with the emulator executable.

Changing Default Resolution for Linux SDK

To change the resolution add the following line to the DirectFB resource file ~/.directfbrc.

```
mode=720x400
```

Another way to change the resolution is to add the following parameters in the command line:

```
DFBARGS=mode=720x400
```

For example, the command to execute an application with a specific screen resolution of 640x480 could be:

```
/usr/local/Oracle_JavaME_Embedded_Client/1.0/
emulator-platform/bin/cvm
-Xdevice:HDTV -DFBARGS=mode=640x480 -Xms48m
-cp /home/dawn/NetBeansProjects/test/dist/test.jar test.Main
```
Stop a Running VM

If you want to stop the VM emulator (while running in debug mode or not) it is not sufficient to press the red stop key in NetBeans. Pressing the stop key only stops the NetBeans build. There are several ways to stop the VM application:

- On your application’s window, press the X button.
- Focus on your application’s window and press CTRL+C.
- Ubuntu Linux: If the above methods don’t kill the window, you can stop a running VM from the command line as follows:

```bash
for pid in `ps aux | grep Oracle_JavaME_Embedded_Client/1.0/emulator-platform/bin/cvm | grep -v grep | awk '{print $2}'`
    do kill -KILL $pid; done

for pid in `ps aux | grep Oracle_JavaME_Embedded_Client/1.0/binaries/bin/cvm | grep -v grep | awk '{print $2}'`
    do kill -KILL $pid; done
```

- ARM, PPC, or MIPS Linux: To stop a running VM, use the following command:

```bash
for pid in `ps aux | grep Oracle_JavaME_Embedded_Client/1.0/binaries/bin/cvm | grep -v grep | awk '{print $2}'`
    do kill -KILL $pid; done
```