WebSphere Micro Environment
Toolkit for Palm OS Developers
"Debugging Setup with Sun JDB Walkthrough"
March 2004

This is a step-by-step walkthrough on how to enable basic debugging using the utilities included in the WebSphere Micro Environment Toolkit for Palm OS Developers (known from here as just "Toolkit").

WebSphere Micro Environment for the Palm OS supports the Java Wire Debug Protocol (JDWP) which allows any compliant debugger to connect to it and control application execution. The Toolkit provides the J9 Debug Server, which hosts an application's symbol "metadata" off device to reduce the burden of debugging on the limited resource embedded environment.

Debugging can be done using a palmOne device Simulator running on a Microsoft Windows development workstation, or on an actual device that is able to handle incoming TCP/IP connections, such as a Tungsten C. On device debugging may not be possible on a Treo 600, as the carrier who operators the WAN may have inbound firewalls in place.

If you have any local firewalls or VPN clients running on your development workstation, you may want to first disable or disconnect them. This setup requires for two applications to act as servers on ports 8888 and 8096.

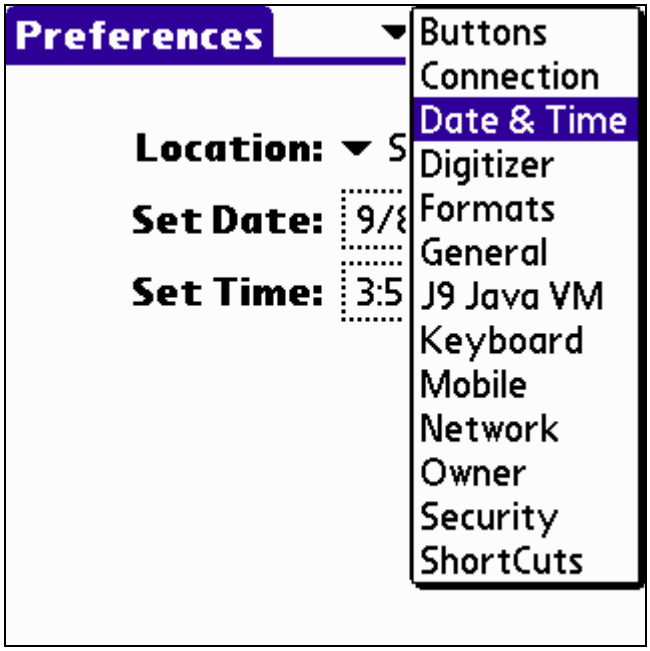
1) Install and run a palmOne Tungsten or Treo Simulator (available through http://pluggedin.palmone.com)

2) Install the necessary runtime files for the Simulator. The Simulator requires the files from the “prc/IA32” directory in the Toolkit distribution. The DLL files must be copied to the Simulator home directory. The PRC files are installed by dragging the file onto the Simulator, or right-clicking on the Simulator.

Please see the “UserGuide.pdf” file in the Toolkit distribution for more information.

3) Install your MIDlet Suite, which should be converted to a Palm PRC application using the Toolkit’s JarToPRC tool.

4) Select the “System” Launcher screen using the upper right-hand corner drop-down, and run the “Prefs” application. In the “Prefs” application, use the upper right-hand corner drop-down to select “J9 Java VM”.



5) Enable the debug settings for your application by doing the following:

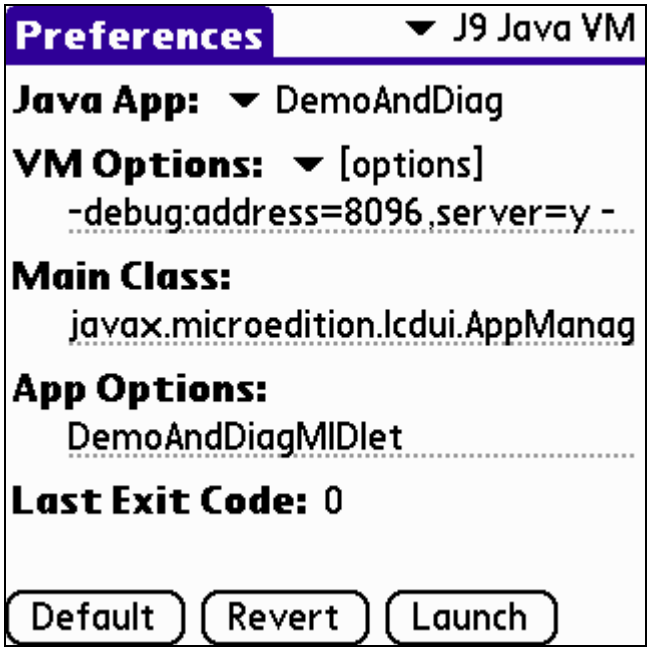
i. Select the name of your application in the “Java App” dropdown
ii. Enter the following text in the “VM Options” textfield:

```
-debug:address=8096,server=y –Xrdbginfo:localhost:8888 –
XstdoutOnScreen
```

(the last argument “-XstdoutOnScreen” is optional, but helpful!)

iii. Enter the following text into the “Main Class:” textfield:
`javax.microedition.lcdui.AppManager`

iv. Enter the complete class name (with package) of your MIDlet in the “App Options” field. In the example below, it is “`DemoAndDiagMIDlet`” which is in the default package.



6) Save your current settings into a Palm Emulator Profile, by right clicking on the emulator and select “Save”. This will save time so you don’t have to re-enter this text again. Currently, if you install a new build of the PRC, you will have to enter this information again.

7) Start the J9 Debug Symbol Server “j9dbgserv” which is included in the “bin” directory of the Toolkit. You must pass the path to the *.sym files that were generated by the JarToPRC application when you converted your JAR and JAD file to a PRC. The argument is “-symfiles:” with the path to each file following, and each entry separated by a semi-colon. You can optionally place each path entry in quotes, which will handle any spaces in the directory or file names.

```
j9dbgserv -symfiles:"DemoAndDiag.sym";"DemoAndDiag1.sym"
Loaded symbol file: DemoAndDiag.sym
Loaded symbol file: DemoAndDiag1.sym
<Waiting for connection from VM on port: 8888>
```

The j9dbgserv should startup, and wait for the VM to connect to it, as shown above.

8) Next start your application by pressing the “Launch” button on the J9Prefs panel, or through choosing the icon on the Palm OS Launcher.

The console output of the j9dbgserv application should then print out the following message:

```
<VM connected from: localhost/127.0.0.1>
```


9) Start the Java Debugger reference application included by Sun in the Java Development Kit. Enter the value for the “-connect” argument as shown below.

```
F:\jdk\bin>jdb -connect
com.sun.jdi.SocketAttach:hostname=localhost,port=8096
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
Initializing jdb ...
>
```

The screen on the Simulator should print out a message saying “debugger connected…” as shown in the screenshot below.

You now have the three components connected. The WEBSPHERE MICRO ENVIRONMENT Virtual Machine is connected to the j9dbgserv to do symbol lookup (which enables breakpoints, locals lookup, etc.). The JDB is connected to the MIDlet running on the WEBSPHERE MICRO ENVIRONMENT that will allow you to step through the application and inspect state at different points you choose.

10) The log below demonstrates a basic debugging session using the JDB, and in particular shows the commands that are enabled currently. Not all JDWP functions are supported in this release, but they should be supported in an update in the near future.

```
VM Started: No frames on the current call stack

main[1] stop at DemoAndDiagMIDlet:54
Deferring breakpoint DemoAndDiagMIDlet:54.
It will be set after the class is loaded.
main[1] next
> Set deferred breakpoint DemoAndDiagMIDlet:54

Breakpoint hit: "thread=Thread-2", DemoAndDiagMIDlet.run(),
line=54 bci=40

Thread-2[1] locals
Method arguments:
Local variables:
dc = instance of com.palm.midp.ui.draw.DrawCanvas(id=2468448)
is = instance of com.palm.midp.ui.draw.ImageShape(id=2468452)
Thread-2[1] dump dc
 dc = {
    shapes: instance of java.util.Vector(id=2468456)
    currShape: null
    offscreen: instance of
javax.microedition.lcdui.Image(id=2468460)
    drawColor: 0
    mode: 1
    fnt: instance of javax.microedition.lcdui.Font(id=2468464)
    POINT: 1
    RECT: 2
    LINE: 3
    TEXT: 4
    javax.microedition.lcdui.Canvas.fFullScreen: false
    javax.microedition.lcdui.Canvas.fRepaintManager: instance of
javax.microed
ion.lcdui.RepaintManager(id=2468468)
    javax.microedition.lcdui.Canvas.DOWN: 6
    javax.microedition.lcdui.Canvas.FIRE: 8
    javax.microedition.lcdui.Canvas.GAME_A: 9
    javax.microedition.lcdui.Canvas.GAME_B: 10
    javax.microedition.lcdui.Canvas.GAME_C: 11
    javax.microedition.lcdui.Canvas.GAME_D: 12
    javax.microedition.lcdui.Canvas.KEY_NUM0: 48
    javax.microedition.lcdui.Canvas.KEY_NUM1: 49
    javax.microedition.lcdui.Canvas.KEY_NUM2: 50
    javax.microedition.lcdui.Canvas.KEY_NUM3: 51
    javax.microedition.lcdui.Canvas.KEY_NUM4: 52
    javax.microedition.lcdui.Canvas.KEY_NUM5: 53
    javax.microedition.lcdui.Canvas.KEY_NUM6: 54
    javax.microedition.lcdui.Canvas.KEY_NUM7: 55
    javax.microedition.lcdui.Canvas.KEY_NUM8: 56
```



```
    javax.microedition.lcdui.Canvas.KEY_NUM9: 57
    javax.microedition.lcdui.Canvas.KEY_POUND: 35
    javax.microedition.lcdui.Canvas.KEY_STAR: 42
    javax.microedition.lcdui.Canvas.LEFT: 2
    javax.microedition.lcdui.Canvas.RIGHT: 5
    javax.microedition.lcdui.Canvas.UP: 1
    javax.microedition.lcdui.Canvas.KEYCODE_A: 65
    javax.microedition.lcdui.Canvas.KEYCODE_B: 66
    javax.microedition.lcdui.Canvas.KEYCODE_C: 67
    javax.microedition.lcdui.Canvas.KEYCODE_D: 68
    javax.microedition.lcdui.Canvas.KEYCODE_0: 48
    javax.microedition.lcdui.Canvas.KEYCODE_1: 49
    javax.microedition.lcdui.Canvas.KEYCODE_2: 50
    javax.microedition.lcdui.Canvas.KEYCODE_3: 51
    javax.microedition.lcdui.Canvas.KEYCODE_4: 52
    javax.microedition.lcdui.Canvas.KEYCODE_5: 53
    javax.microedition.lcdui.Canvas.KEYCODE_6: 54
    javax.microedition.lcdui.Canvas.KEYCODE_7: 55
    javax.microedition.lcdui.Canvas.KEYCODE_8: 56
    javax.microedition.lcdui.Canvas.KEYCODE_9: 57
    javax.microedition.lcdui.Canvas.KEYCODE_STAR: 42
    javax.microedition.lcdui.Canvas.KEYCODE_POUND: 35
    javax.microedition.lcdui.Canvas.KEYCODE_UP: -1
    javax.microedition.lcdui.Canvas.KEYCODE_DOWN: -2
    javax.microedition.lcdui.Canvas.KEYCODE_LEFT: -3
    javax.microedition.lcdui.Canvas.KEYCODE_RIGHT: -4
    javax.microedition.lcdui.Canvas.KEYCODE_FIRE: -5
    javax.microedition.lcdui.Canvas.KEYCODE_INVALID: 0
    javax.microedition.lcdui.Canvas.HORIZONTAL_TAB_CHARACTER: -10
    javax.microedition.lcdui.Canvas.BACKSPACE_CHARACTER: -11
    javax.microedition.lcdui.Canvas.DELETE_CHARACTER: -12
    javax.microedition.lcdui.Canvas.NULL_CHARACTER: -13
    javax.microedition.lcdui.Canvas.UNKNOWN_CHARACTER: -14
    javax.microedition.lcdui.Displayable.fCommands: null
    javax.microedition.lcdui.Displayable.fCommandListener: null
    javax.microedition.lcdui.Displayable.fTitle: null
    javax.microedition.lcdui.Displayable.fDisplayReadyTitle: null
    javax.microedition.lcdui.Displayable.fTicker: null
    javax.microedition.lcdui.Displayable.fPeer: null
    javax.microedition.lcdui.Displayable.TYPE_CANVAS: 0
    javax.microedition.lcdui.Displayable.TYPE_LIST: 1
    javax.microedition.lcdui.Displayable.TYPE_FORM: 2
    javax.microedition.lcdui.Displayable.TYPE_ALERT: 3
    javax.microedition.lcdui.Displayable.TYPE_TEXTBOX: 4
}
Thread-2[1] dump is
 is = {
    img: instance of javax.microedition.lcdui.Image(id=2468472)
    clip: instance of com.palm.midp.ui.draw.Rectangle(id=2468476)
    com.palm.midp.ui.draw.Shape.color: 0
    com.palm.midp.ui.draw.Shape.x: 0
    com.palm.midp.ui.draw.Shape.y: 0
}
Thread-2[1] dump is.clip
 is.clip = {
    width: 160
    height: 144
```



```
    com.palm.midp.ui.draw.Shape.color: 0
    com.palm.midp.ui.draw.Shape.x: 0
    com.palm.midp.ui.draw.Shape.y: 0
}
Thread-2[1] dump is.img
 is.img = {
    fHandle: 723460
    fWidth: 160
    fHeight: 144
    fDepth: 8
    fMutable: false
    fDisposeBitmap: true
    fCachedGraphics: null
    fCanCache: true
    fBitmapWindow: 0
    gImageCache: instance of java.util.Vector(id=2468480)
    DEFAULT_DEPTH: 8
}
Thread-2[1] next
➢  cont
```

The commands demonstrated in the JDB console output above, show the basic process of connecting to the WEBSPHERE MICRO ENVIRONMENT and J2ME app running on the emulator, setting breakpoints, stepping through execution, and printing out the state of the application at each step. Collectively, these allow you, the developer, to debug your application while it executes in the actual Palm MIDP/CLDC runtime environment.

In the future, this debugging capability will be enabled and integrated for use with other Java Integrated Development Environments(IDEs) that support the Java Debug Wire Protocol (JDWP).

