How to configure Java ODD over WLAN with Eclipse and NetBeans IDEs

Overview

05 Apr 2009

On-device debugging feature (ODD) can be used for debugging a MIDlet running on a phone by using a bearer (in this case, WLAN) for sending and receiving information between PC and the phone. This feature has been tested with NetBeans 6.5 and Eclipse 3.4 IDEs, also other NetBeans 6.x and Eclipse 3.x versions should work just fine.

By using ODD, it is possible to control the application flow by entering breakpoints, follow the variable values in the IDE's debugging view, etc. It is highly recommended to use public (non-secured) wireless LAN network, there are known problems in case of secured networks. Simple way is to use a WLAN router and make connection between the PC and the phone.

Note: This article explains, how ODD is done by using the S60 5th Edition SDK 1.0 and the EcmtAgent tool included with it. The EcmtAgent tool was updated in this SDK and its usability was improved.

How to configure Java ODD over WLAN with Eclipse and NetBeans IDEs:

1. Install "EcmtAgent.sis" on to the device
2. Connect the device to a WLAN network
3. Launch EcmtAgent application on the device
4. Edit system IP address and port number
5. Check the device IP address
6. Connect the PC to the same WLAN network, which was used with the device
7. Open Eclipse IDE or NetBeans IDE on the PC
8. Configure the IDE debug settings
9. Start the MIDlet on the device
10. Start debugging from the IDE

Step-by-step instructions

Step 1: Install "EcmtAgent.sis" on to the device

This sis file can be found in the following path of the installed SDK <your SDK installation folder>\S60tools\Ecmt. By default the path is "C:\S60\Devices\S60_5th_Edition_SDK_v1.0\S60tools\Ecmt".

Step 2: Connect the device to a WLAN network

In Nokia 5800 XpressMusic, select "Menu" -> "Settings" -> "Connectivity" -> "Wireless LAN" -> "Open". When the suitable WLAN network is shown in the list, select it and select "Connect".

Step 3: Launch EcmtAgent application on the device

The installed application can be found in the device under "Menu" -> "Applications". Make sure, that the EcmtAgent application is listening WLAN. If it is not, select "Settings" from Options menu. Change the Bearer to WLAN. Go back to main screen. The device might ask to select the WLAN network to be used, if several of them are available. If an error message "Engine error -18" is shown on the EcmtAgent screen, it means, that there are problems with the WLAN connection.

Step 4: Edit system IP address and port number

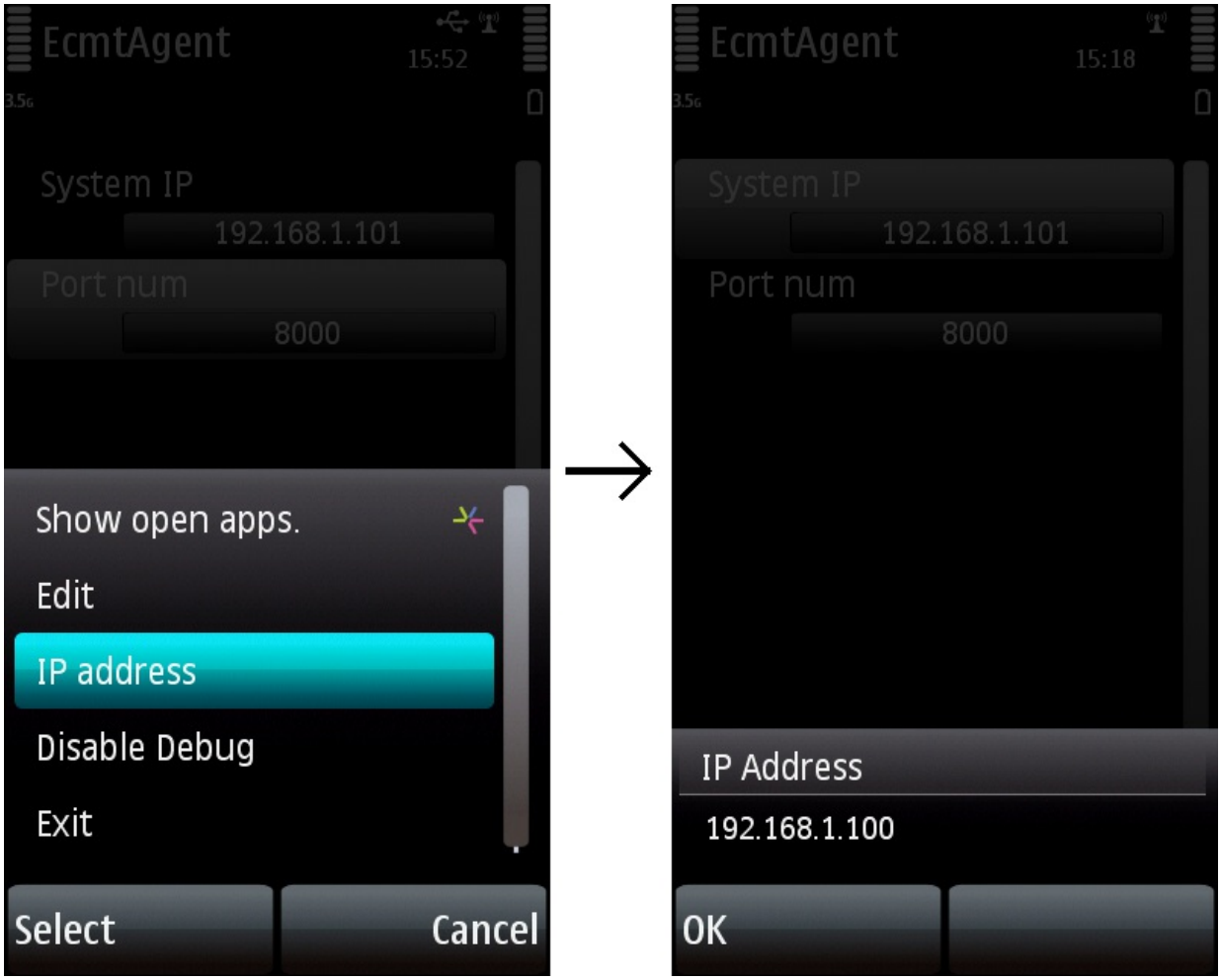
Once the application is launched and it is listening to the correct WLAN network, select "WLAN Java ODD" from the Options menu as shown:

The application switches to a new view where the user can enter the System IP address and port number. The system IP address is the PC's IP address, to which the device should connect. It can be found for example by opening Windows command prompt and entering "ipconfig" to it. The correct IP address is commonly shown under "Ethernet adapter Wireless Network Connection:".

Edit the system IP address setting box and enter the PC IP address where IDE is running. After this edit the port setting box and enter any free port available on the PC. The used ports can be listed by opening Windows command prompt and entering "netstat –a". This command lists all the active connections. Used port numbers are shown in the Local address column separated by colon.

Step 5: Check the device IP address

To get the IP address of the device select "Options" -> "IP address". This will give you the IP address of the device. Make note of this IP address. Note, that the device should be connected to the WLAN before the step 5 is executed.



You can exit the EcmtAgent application now(, but it can be left running in the background too).

Step 6: Connect the PC to the same WLAN network, which was used with the device

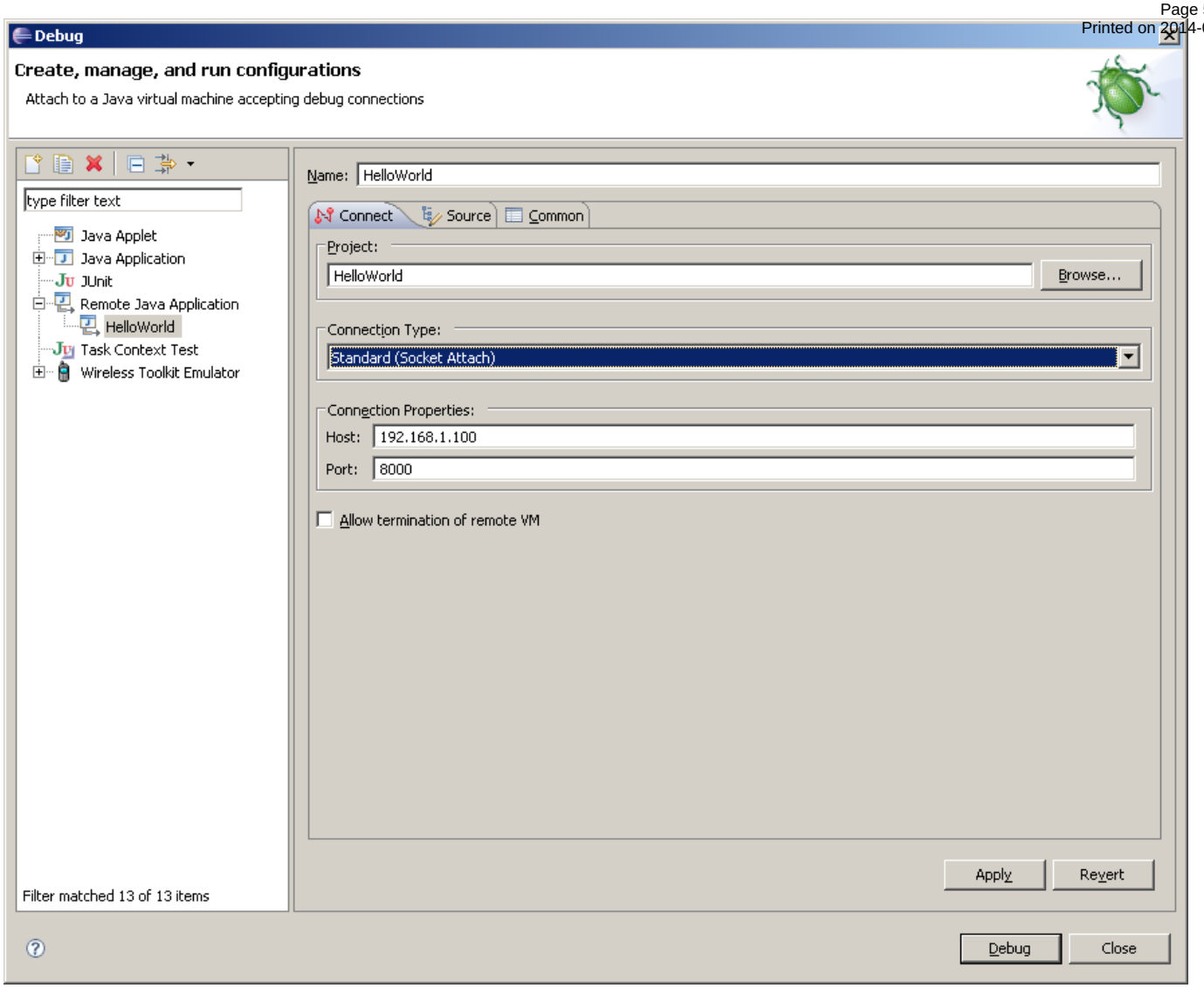
The details of this step vary depending on the PC WLAN configuration.

Step 7: Open Eclipse IDE or NetBeans IDE on the PC

Open the Project which has to be debugged on the target device. Install the JAD and JAR files of the project on to the device. In case of Eclipse IDE, use for example PC Suite Application installer for this. In NetBeans IDE use its deployment feature for the installation.

Step 8: Configure the IDE debug settings

In Eclipse open the Debug dialogue box of the project: right-click the project in the Package Explorer, select "Debug As" -> "Open Debug Dialog…". Create a new instance of "Remote Java Application" as shown:



Once a new instance is created, provide the IP address of the target device in the "Connection Properties" -> Host. This IP address should be same as what was found out in Step 5. Provide also the port number as shown in the image below. This port number should be same as what was given in Step 4.

Step 9: Start the MIDlet on the device

Run the MIDlet application on the device first. The device prompts a dialogue box saying that it is using modified VM arguments. Then it checks the WLAN connection on the device. Select a common WLAN connection where PC is accessible.

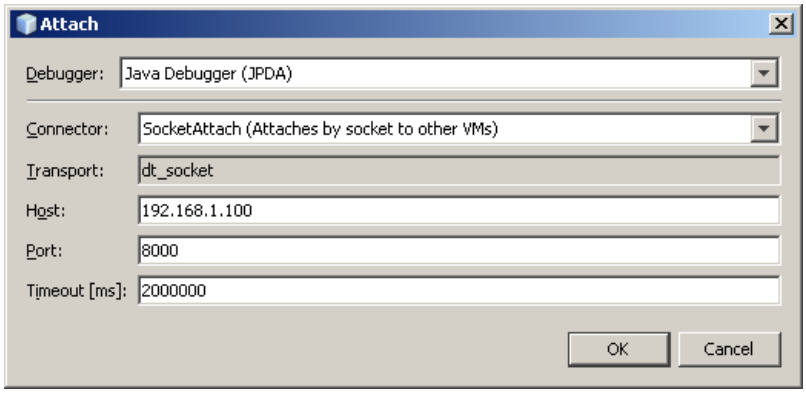
Note: When EcmtAgent has been used for getting the device IP address, it also modifies the VM arguments. When the arguments have been changed, launching MIDlets normally doesn't work. For resetting the VM arguments, select in EcmtAgent: "Options" -> "WLAN Java ODD" -> "Disable Debug" and close the EcmtAgent.

Step 10: Start debugging from the IDE

Once the connection is successful, click the "Debug" button on Eclipse IDE debug dialogue box. Once the PC sets up connection on the WLAN, it changes its view to debug mode. Now the user will be able to debug the application on the IDE.

Note: The debugger timeout must be large enough. In Eclipse IDE it can be set in "Window" -> "Preferences" -> "Java" -> "Debug". Default value is usually 3000 ms. It can be increased to, let's say 2000000 ms, just to be sure, that the timeout is not a problem.

In case of NetBeans IDE the debugging details are added in the Debug menu: "Debug" -> "Attach debugger…", as shown below:



Note: The timeout should be large enough. Here 2000000 ms is used, it should be more than enough. Note also, that in case of NetBeans the MIDlet should be started before this step. Now, when "OK" is pressed, debugging process starts and the MIDlet is launched in the device.

See also

- S60 5th Edition SDK 1.0
- How to get System.out output from a MIDlet and save it to a file in S60 devices

