S60 3rd Edition SDK Supporting Feature Pack 1, for MIDP, User's Guide

Version 1
November 17th, 2006

S60 platform

Contents

1 Introduction

1.1 Introduction

Thank you for installing the S60 3rd Edition Software Development Kit for Symbian OS, Supporting Feature Pack 1, for MIDP.

With the SDK you can develop and implement MIDP applications for S60 platform smartphones on your PC. Together with an Integrated Development Environment (IDE), the SDK provides all the needed functionality for this, including Application Programming Interfaces (APIs), sample code and documentation needed for developing new S60 MIDlets, that is, applications that conform to the Mobile Information Device Profile (MIDP).

1.2 SDK contents

Once you have downloaded and installed the SDK, the following components and functionality are installed on your PC:

- S60 software libraries and Application Programming Interfaces (APIs)
- S60 emulator
- SDK documentation
- Example MIDlets (source code and documentation)

Software libraries and APIs provide the programming interfaces that you need in developing Java applications for S60 smartphones. For API documentation, please refer to the API documentation (Javadocs), located in `<S60_SDK_installation_directory>\docs`.

The S60 emulator enables you to view and test applications on your PC before installing them to a real device. The emulator mimics the operation of an application on a real phone so accurately that application development can be done even before the required hardware, that is, an S60 smartphone, is available. For more detailed information on the S60 emulator, please refer to the **Emulator Guide** in the **Tools and Utilities** section.

The SDK documentation includes information and instructions on how to use the SDK, API Reference documentation (JavaDocs) and Java implementation Notes. For more information on the documentation provided with the SDK, please refer to Help contents on page 0 .

A number of S60 example MIDlets are delivered with the S60 SDK. The MIDlets demonstrate the variety of different applications that you can run on S60 devices. You can build and run the MIDlets on the S60 emulator as well as use them in creating your own applications. The S60 example MIDlets are located by default in the following folder after SDK installation:

```
<S60_SDK_installation_directory>\S60examples\
```

1.3 What's new in this release

The following new features are introduced in the S60 3rd Edition SDK for Symbian OS Supporting Feature Pack 1, for MIDP:

- **New APIs:**
  - SATSA-APDU optional package for JSR-177 SATSA
  - JSR Advanced Multimedia Supplements for J2ME (partially beta-level implementation, see Release Note for more details)
  - JSR-226 Scalable 2D vector graphics
- **Updated APIs:**
  - JSR-184 Mobile 3D Graphics (m3g) API for J2ME v1.1
  - JSR-135 MMAPI
  - JSR-75 PIM API
  - JSR-120 WMA

For more information on APIs and their documentation, please refer to Javadocs.

- **New examples:**
  - golfhelper: a JSR 179 Location API example
  - mms: a JSR 205 MMS API example
  - security: a JSR 177 SATSA for Java Platform, Micro Edition example
  - sip: a JSR 180 SIP example
  - svg: a JSR-226 M2G API example

For more information on example MIDlets and their documentation, please refer to S60 example MIDlets.

1.4 Supported interfaces, tool set-ups and accessories

The S60 3rd Edition SDK for Symbian OS Supporting Feature Pack 1, for MIDP has been tested on Microsoft Windows 2000 (SP4) and Microsoft Windows XP (SP2). These are currently the only supported operating systems - other Microsoft Windows operating systems may work, but have not been used in testing.

Interfaces

S60 3rd Edition SDK for Symbian OS Supporting Feature Pack 1, for MIDP supports usage of Sun's Universal Emulator Interface (UEI) interfaces (OTA and D-flags are not supported by the SDK). Consequently, you can use any IDE supporting the UEI together with the S60 SDK to develop Java MIDP applications.

Java APIs

The S60 SDK supports the following Java APIs:

- Advanced Multimedia Supplements for J2ME (JSR-234) - partially beta-level implementation (see *SDK Release Note* for more details)
- Bluetooth API 1.0 (JSR-82) (including Push support and Obex support)
- FileConnection Optional Package 1.0 (JSR-75) of PDA Optional Packages for the Java™ Platform, Micro Edition (Java™ ME)
- J2ME Web Services API (JSR-172)
- Java Connected Limited Device Configuration 1.1 (JSR-139)
- Java Mobile Information Device Profile 2.0 (JSR-118)
- Java Technology for the Wireless Industry 1.0 (JSR-185)
- Location API for J2ME (JSR-179)
- Mobile 3D Graphics API for J2ME (JSR-184)
- Mobile Media API 1.1 (JSR-135)
- Nokia UI API 1.1
- Personal Information Management (PIM) Optional Package (JSR-75) of PDA Optional Packages for the Java™ Platform, Micro Edition (Java™ ME)
- SATSA-APDU optional package (for JSR-177 SATSA)
- Scalable 2D vector graphics (JSR-226)
- Security and Trust Services API for J2ME (JSR-177). **Note:** The SDK supports SATSA-PKI, SATSA-CRYPTO and SATSA-APDU optional packages, but the emulator only supports the first and the second.
- SIP API for J2ME (JSR-180)
- Wireless Messaging API 2.0 (JSR-205)

For more information on the APIs and their documentation, please refer to Javadocs.

Tool Set-ups

S60 SDK supports the following tool setups:

- Carbide.j 1.5 (with Eclipse)
- Eclipse 3.1.2 (or newer)
- NetBeans 5.0 Mobility Pack
- IBM WebSphere Studio Device Developer 5.7.1
- Nokia Mobile Internet Toolkit 4.1 (NMIT 4.1).
- Nokia Developer's Suite for MMS v1.1 (NDSforMMS v1.1)

Supported Accessories

S60 3rd Edition SDK for Symbian OS Supporting Feature Pack 1, for MIDP supports the following accessories:

Bluetooth cards and dongles

- TDK BRBLU04
- Brainboxes BL-500
- Casira (HCI BCSP & H4)
- Cards supporting HCI BCSP
- Wrapper for Bluetooth USB dongles (with CSR chipset) at H4 mode

Infrared devices

- Extended Systems ESI-9680 RS-232 IR pod (or compatible)

USB versions

- v.1.0, v.1.1 and v.2.0.

Ethernet network cards

- Ethernet network cards complying with IEEE 802.3 standards (Ndismedium802_3) .

Supported Languages

The emulator supports application development in the following languages:

- English
- Chinese
- Japanese
- Thai

1.5 Javadocs

Software libraries and APIs provide the programming interfaces needed in developing Java applications for S60 smartphones. Documentation (Javadocs) is provided for each API provided with the S60 3rd Edition SDK for Symbian OS Supporting Feature Pack 1, for MIDP.

The Javadocs are located in `<S60_SDK_installation_directory>\docs\` (to open the Javadoc of an API, go to `<S60_SDK_installation_directory>\docs\<API_name>\index.html`).

Javadocs are provided for the following APIs:

- Advanced Multimedia Supplements API (JSR-234)
- Bluetooth API (JSR-82)
- Connected Limited Device Configuration 1.1 (CLDC, JSR-139)
- FileConnection Optional Package 1.0

- Location API for J2ME (JSR-179)
- MID Profile (JSR-118)
- Mobile 3D Graphics API (M3G) for J2ME
- Mobile Media API (JSR-135)
- Nokia UI API v1.1
- PIM Optional Package 1.0 Specification
- Security and Trust Services API (SATSA) (JSR-177)
- Scalable 2D Vector Graphics API for J2ME (JSR-226)
- SIP API for J2ME (JSR-180)
- Web Services jaxp (JSR-172)
- Web Services jaxrpc (JSR-172)
- Wireless Messaging API (WMA) 2.0
- Wireless Messaging API (JSR-120)

Note: The SDK Help also provides **Java Implementation Notes**, which provide information on the S60-specific implementation details of the APIs, including a list of supported and unsupported features as well as any class-level differences existing compared to reference implementation.

1.6 S60 example MIDlets

The S60 3rd Edition SDK for Symbian OS, Supporting Feature Pack 1, for MIDP installation package includes the following example MIDlets to demonstrate the use of the supported APIs:

- **AMMSMIDlet** - demonstrates how to use environmental 3D audio in applications. The MIDlet implements the AMMS API defined in JSR 234. Located in the `S60examples\AMMSMIDlet` directory.
- **btl2capEcho** - a JSR 82 Bluetooth L2CAP API example, located in the `S60examples\btl2capEcho` directory.
- **btsppEcho** - a JSR 82 Bluetooth SPP API example, located in the `S60examples\btsppEcho` directory.
- **capitalclient** - a JSR 172 Web Service client example, located in the `S60examples\capitalclient` directory.
- **capitalservlet** - a JSR 172 Web Service client example, located in the `S60examples\capitalservlet` directory.
- **chat** - a JSR 120 example, located in the `S60examples\chat` directory.
- **devicecap** - a JSR 135 MMA API example, located in the `S60examples\devicecap` directory.
- **fileconnection** - a JSR 135 MMA API example, located in the `S60examples\fileconnection` directory.
- **filehandler** - a JSR 75 example, located in the `S60examples\filehandler` directory.
- **golfhelper** - a JSR 179 Location API example, located in the `S60examples\golfhelper` directory.
- **hawk** - a JSR 118 MIDP 2.0 GameCanvas example, located in the `S60examples\hawk` directory.
- **helloworldplus** - a simple JSR 118 MIDP 2.0 example, located in the `S60examples\helloworldplus` directory.
- **m3g** - a JSR 184 Mobile 3-D Graphics API example, located in the `S60examples\m3g` directory.
- **mms** - a JSR 205 MMS API example, located in the `S60examples\mms` directory.

- **personalcontroller** - a JSR 75 example, located in the `S60examples\personalcontroller` directory.
- **pki** - demonstrates how to use a smart card to digitally sign data and manage user certificates, it implements the PKI API in the Security and Trust Services API 1.0 defined in JSR 177. Located in the `S60examples\pki` directory.
- **push** - a JSR 118 MIDP push example, located in the `S60examples\push` directory.
- **security** - a JSR 177 SATSA for Java Platform, Micro Edition example, located in the `S60examples\security` directory.
- **sheepdog** - a JSR 118 MIDP 2.0 GameCanvas example, located in the `S60examples\sheepdog` directory.
- **shopping** - a JSR 118 MIDP RMS example, located in the `S60examples\shopping` directory.
- **sip** - a JSR 180 SIP example, located in the `S60examples\sip` directory.
- **svg** - a JSR-226 M2G API example, located in the `S60examples\svg` directory.
- **XMLParser** - demonstrates how to parse a file written in XML format. The MIDlet implements Web Services support defined in JSR 172. Located in the `S60examples\XMLParser` directory.

For instructions on how to run the example MIDlets on the S60 emulator, please refer to the separate example MIDlets documentation set located in `<S60_SDK_installation_directory>\S60examples\doc-files\examples.html`

1.7 S60 Platform

S60 Platform is a complete package of applications, user interface and development tools built upon Symbian OS technology. It has been designed to run on various models of different manufacturer's devices. In other words, an application developed for S60 Platform will run smoothly on all devices that run the same platform.

The S60 User Interface (UI) has been specifically designed for easy, one-handed use. From the user's point of view, probably the most important feature of the platform is its user interface: A large color screen (several resolutions) and the various input keys (two soft keys, five-way navigator and several dedicated keys). The principle of one-handed use and the large color screen also enables application developers to present attractive content and provide easy navigation. It is easy for any user with experience of mobile phones to grasp the basic idea of the UI.

Figure 1: S60 Platform user interface

There are also a variety of applications to be found on S60 Platform. Important amongst these are the advanced smartphone telephony applications, Personal Information Management (PIM) applications (for example, Phonebook, Calendar, Photo Album, etc.), messaging, browsing, e-mail and an installation engine that allows the user to add or remove applications to and from the platform, either via PC Connectivity or Over-the-Air downloads.

S60 Platform also provides support for applications written in the Java programming language. The Java platform implemented on S60 devices is the Java 2 Platform Mobile Edition (J2ME™), which is designed for small mobile devices, such as smartphones.

Any Java application can easily be delivered over the Internet, or any network, without operating system or hardware platform compatibility issues. Java technology components run on any kind of compatible device that supports the Java platform. Note, however, that using the Nokia-specific Nokia UI API included in the SDK affects the portability of your application.

For more detailed information about developing Java™ applications for teh S60 Platform, please refer to the Getting started with Mobile Java™ section of the SDK Help

1.8 Symbian OS 9.2

Symbian OS v. 9.2 is an operating system optimized for mobile terminals, such as communicators and phones, and it is the underlying system upon which the S60 platform has been built. Symbian OS provides a reliable environment as it has been designed so that user data is never lost and the device running the operating system will never have to be rebooted.

Symbian OS is thoroughly object-oriented; all the system objects from applications to interrupt handlers are defined as C++ classes. This contributes to its flexibility, efficiency, and ability to re-use segments of the code. Clearly defined Application Programming Interfaces (APIs) allow the developer community, including terminal vendors, to create applications that can be easily downloaded, installed, and run natively on the terminal Symbian OS. Symbian C++ Application Programming Interfaces (APIs) enable extremely efficient multitasking and memory management. Processor and memory-intensive operations, such as context switching, are minimized.

Symbian OS, upon which the S60 platform is built, provides APIs enabling the utilization of a multitude of technologies, such as Bluetooth, graphics, infrared, multimedia, messaging, networking, and telephony.

Key features of Symbian OS 9.2

The following list describes key features of the Symbian OS 9.2:

- Rich suite of application services - the suite includes services for contacts, scheduling, and messaging, OBEX for exchanging appointments (vCalendar) and business cards (vCard); integrated APIs for data management, text, clipboard and graphics.
- Realtime - a realtime, multithreaded kernel provides the basis for a robust, power-efficient and responsive phone.
- Hardware support - supports latest CPU architectures, peripherals and internal and external memory types.
- Messaging - enhanced messaging (EMS) and SMS; internet mail using POP3, IMAP4, SMTP and MHTML; attachments.
- Multimedia - audio and video support for recording, playback and streaming; image conversion.
- Graphics - direct access to screen and keyboard for high performance; graphics accelerator API; increased UI flexibility (support for multiple simultaneous display, multiple display sizes and multiple display orientation).
- Platform security - proactive system defense mechanism based on granting and monitoring application capabilities through Symbian Signed certification. Infrastructure to allow applications to have private protected data stores. In addition, full encryption and certificate management, secure protocols (HTTPS, SSL and TLS) and WIM framework.
- Communications protocols - wide area networking stacks including TCP/IP (dual mode IPv4/v6) and WAP 2.0 (Connectionless WSP and WAP Push), personal area networking support including infrared (IrDA), Bluetooth and USB; support is also provided for multihoming and link layer Quality-of-Service (QoS) on GPRS and UMTS networks.
- Mobile telephony - Symbian OS v9.2 is ready for the 3G market with support for WCDMA (3GPP R4 and R5 IMS); GSM circuit switched voice and data (CSD and EDGE CSD) and packet-based data (GPRS and EDGE GPRS); CDMA circuit switched voice, data and packet-based data (IS-95 and 1xRTT); SIM, RUIM, UICC Toolkit; other standards can be implemented by licensees through extensible APIs of the telephony subsystem.
- CDMA specific features including CDMA network roaming, third party OTA API, NAM programming mode, CDMA SMS stack, NAI handset identification, interfaces to enable Mobile IP and bridge and router gateway modes of operation.
- International support - supports the Unicode Standard version 3.0.
- Data synchronization - over-the-air (OTA) synchronization support using OMA standards; PC-based synchronization over serial, Bluetooth, infrared and USB; a PC Connectivity framework providing the ability to transfer files and synchronize PIM data.
- Device Management/OTA provisioning - OMA DM 1.1.2 compliant, OMA Client provisioning v1.1.

2 Getting started with Mobile Java™

2.1 About getting started

Getting started with Mobile Java provides you with information on how to start using a MIDP development environment. It goes through the general steps required in developing an application and shows you the default project structure involved in this process. It also includes general designing and testing guidelines, as well as useful check lists for applications. Finally, a Hello World example MIDlet is provided to demonstrate the basics of Java applications and to get you started with your development environment.

The example provided in this document is described using a development environment that includes Eclipse, Carbide.j, and S60 MIDP SDK.

Intended audience

Getting started with Mobile Java is intended for MIDP developers, as well as Java EE and Java SE developers wishing to develop mobile Java applications or services.

Scope

This document assumes a good knowledge of application and service development and the Java programming language. Furthermore, it assumes familiarity with enterprise application development. However, previous knowledge of developing for the mobile environment is not necessary.

For information on Java technology in general, see Java Technology web site at http://java.sun.com/ and Java Mobility Development Center at http://developers.sun.com/techtopics/mobility/.

2.2 Application development process

This section describes the general steps involved in MIDP application development, starting from setting up the project environment and ending in deploying to a target device. Note that in reality all these tasks are not necessarily performed by the same developer and that your process may differ. The following figure illustrates the continuous cycle of creating, compiling, and testing code, involved in the process of developing an application.

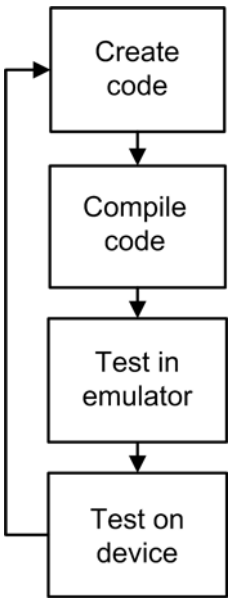


Figure 2: The application testing cycle

The steps involved in developing a MIDP application are:

- Analyze the project

  Determine the requirements, scope, and functionality of your application, as well as the development environment to use.
- Set up project environment

  Set up the project environment in accordance with your IDE requirements. . For information on the default project structure, see section Default project structure.
- Create Java™ source files

  Create or import the source code in Java programming language according to your own processes.
- Compile source files

  Compile the source files into class files according to your own processes.
- Preverify classes

  Preverify the classes according to your own processes.
- Create JAR and JAD files

  Create JAR and JAD files to create MIDlet suite for your application. For information on JAR and JAD files, see section Default MIDP project structure.

  - MIDlets can be deployed using MIDP OTA.
  - MIDlet suites are used for the distribution and installation of services and applications.
- Test code

  Run the application in the emulator and test the code according to your own processes. For information on testing code, see section Designing and testing MIDP applications.
- Sign your application

  Package and sign the application according to your own processes. For information on testing and signing applications for the trusted third party domain, see Java Verified Program at http://javaverified.com/.
- Deploy the application

  Applications can be deployed to devices for testing purposes or distributed to wider audiences.

  - Test applications by deploying them directly on your device, using a USB cable, Bluetooth connection, infrared, or other hardware compatible connection.
  - Distribute the application to multiple devices through any one of the standard methods available. OTA technology can also be used.

  For information on deploying MIDlets, see section Deploying MIDlets to a device.

2.3 Default MIDP project structure

This section describes the main file and folder types involved in the different phases of application development. The following figure presents a view of a HelloWorld application project structure. Depending on the actual content of your project, there may be differences.

Note: This section deals with the issue based on a development environment that includes Eclipse and S60 MIDP SDK. The project structure may have some variations in different environments, but the basic principles are still the same.

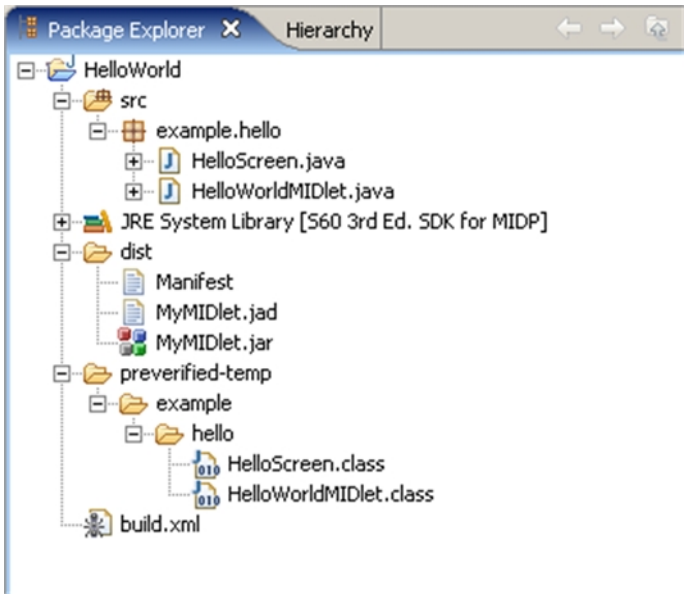


Figure 3: Default Eclipse project structure

Setting up the project environment

This phase includes setting up a working environment for the project.

- `src` folder

  This folder is meant for the source Java™ files of the project.
- `res` folder

  This folder is meant for all the possible resource files that are needed in the final deployment package, such as figures, icons, and documents. These are not used in the Hello World example application, but could be part of other similar applications.
- `bin` folder

  This folder is meant for the compiled class files.
- `dist` folder

  This folder is meant for the JAR an JAD files.

Creating and compiling source files

This phase includes creating and compiling source files, as well as creating or importing other necessary resources.

- `*.java` files

  Java files (`*.java`) are the source files of the application created by the developer. Java files are located in the `src` folder.
- Resource files

  Resource files, if they exist, can include image and document files that need to be enclosed in the JAR file and are required by the application. Resource files are located in the `res` folder.
- `JRE System Library`

  This folder includes all the required libraries provided by the SDK, for example, `JRE System Library [S60_3rd_MIDP_SDK]`.
- `*.class` files

Java class files (`*.class`) are source files compiled into bytecode format so that they can be interpreted and executed by the Java Virtual Machine (JVM). Java class files are located in the bin folder and will be enclosed in a JAR file.

Creating MIDlet suites

This phase includes combining the related files into a JAR file and creating an accompanying JAD file for a MIDlet suite, which can be deployed onto a mobile device.

The JAR (Java Archive Repository) file format is used for combining several files and resources into one JAR file and can be subsequently transferred to a mobile device as a single package. This format also supports file compression and digital signatures. The Hello World example application includes one application JAR file. An application package also contains the following files:

A JAR file includes a Manifest file (`MANIFEST.MF`), which is a descriptor file. This file is an essential part of the signed MIDlet model in the MIDP 2.0 specification and is mandatory. Manifest files can be created by defining the appropriate information in the while creating the JAR file, or by using an existing manifest file.

The following manifest file attributes are mandatory:

- `MIDlet-Name`
- `MIDlet-Version`
- `MIDlet-Vendor`

The manifest file or the JAD file must contain the following attributes:

- `MIDlet-<n> for each MIDlet`
- `MicroEdition-Profile`
- `MicroEdition-Configuration`

The following manifest file attributes are optional:

- `MIDlet-Description`
- `MIDlet-Icon`
- `MIDlet-Info-URL`
- `MIDlet-Data-Size`
- `MIDlet-Permissions`
- `MIDlet-Permissions-Opt`
- `MIDlet-Push-<n>`
- `MIDlet-Install-Notify`
- `MIDlet-Delete-Notify`
- `MIDlet-Delete-Confirm`
- Any application-specific attributes that do not begin with `MIDlet-` or `MicroEdition-`

An example manifest file might look like this:

```
MIDlet-Name: MyMIDlet
MIDlet-Version: 0.0.1
MIDlet-Vendor: Nokia
MicroEdition-Profile: MIDP-2.0
MicroEdition-Configuration: CLDC-1.1
MIDlet-1: HelloWorldMIDlet, , example.hello.HelloWorldMIDlet
```

The JAD (Java Application Descriptor) file format is used for managing and describing a MIDlet suite. The file also allows configuration specific attributes that can be used when installing a MIDlet. Each JAR file can come with its own JAD file. A JAD file contains the following mandatory attributes:

- `MIDlet-Name`
- `MIDlet-Version`

- `MIDlet-Vendor`
- `MIDlet-Jar-URL`
- `MIDlet-Jar-Size`

The following JAD file attributes are optional:

- `MIDlet-<n> for each MIDlet`
- `MicroEdition-Profile`
- `MicroEdition-Configuration`
- `MIDlet-Description`
- `MIDlet-Icon`
- `MIDlet-Info-URL`
- `MIDlet-Data-Size`
- `MIDlet-Permissions`
- `MIDlet-Permissions-Opt`
- `MIDlet-Push-<n>`
- `MIDlet-Install-Notify`
- `MIDlet-Delete-Notify`
- `MIDlet-Delete-Confirm`
- MIDlet specific attributes that do not begin with `MIDlet-`

An example JAD file might look like this:

```
MIDlet-Name: MyMIDlet
MIDlet-Version: 0.0.1
MIDlet-Vendor: Nokiay
MicroEdition-Profile: MIDP-2.0
MicroEdition-Configuration: CLDC-1.1
MIDlet-Jar-URL: MyMIDlet.jar
MIDlet-Jar-Size: 1887
MIDlet-1: HelloWorldMIDlet, , example.hello.HelloWorldMIDlet
```

2.4 Designing and testing MIDP applications

This section lists fundamental issues that must be taken into account when designing and testing MIDP applications (MIDlets). It is important to test both the functionality and the usability of an application to ensure that the application is easy and efficient to use, and keeps the end user satisfied.

You should also note that there might be some special issues with devices from certain series. For this reason, please read the possible Known Issues or other similar documents carefully, and test the special issues correspondingly.

2.4.1 Designing for multiple devices

Creating applications for multiple devices produce design challenges that need to be considered at the design phase. The variations between mobile devices mean that checking the implementation details of each device and testing the application on all target devices is mandatory. The following is a list of the main issues in designing for multiple devices:

- Differences between devices

  Different devices run on different developer platforms that include different APIs. Even devices that have the same APIs may have different implementations, including whether all optional packages are implemented or whether a different version of the API is used.

  In addition to the different developer platforms, devices also tend to have a variety of different screen size, memory and other qualities.

- Differences between API implementations

  Even if the devices run on the same developer platform and the same APIs, there may be differences in the implementation of individual mobile device models that may cause problems. For information on all S60 devices, see Device Specifications on the S60 platform Web site at http://www.s60.com/life/s60phones.

- Differences between screen sizes

  Mobile devices tend come in a variety of screen resolutions. This means that an application is more likely to work on devices that have a matching screen resolution to the original target device.

  The MIDP 2.0 specification provides several properties and methods that can be used for querying different aspects of an API implementation. The proper use of these can make porting MIDlets much easier.

  There are two levels of user interface components available in the LCDUI package: high-level APIs and low-level APIs. When high-level APIs and components are used, the MIDlet has a greater portability because logical components are being used, that is, the MIDlet only guides the device on laying out the components. However, even on this level, portability cannot be guaranteed. For example, the aspects of the component layout which are not mandatory cannot be relied to be portable.

  With low-level APIs, the MIDlet draws directly on the screen using lower-level primitives. One important issue concerning these low-level APIs is hard-coding positional information in order to achieve a required layout for a device. This can easily lead to problems as the result can look quite different on other devices. To prevent these type of problems, MIDlets can obtain several variables, such as the canvas or font size, from the device to avoid layout problems. For example, it is a common fault to assume a certain font height and then start the following line of text from a fixed distance below the first line, which may look awkward on some devices.

2.4.2 The testing process

Whether you are developing an enterprise-wide application or a simple cell phone game, the cycle of debugging and testing proceeds similarly — through successive rounds of debugging and testing.

- **Debugging**

  Wireless consumers tend to be less technically oriented than desktop computer users and will be less tolerant when programs fail to work properly. Thorough debugging and testing of wireless applications is extremely important.

  A debugger systematically breaks down your code into distinct sections, monitors the activity, and verifies that the code executes the way it was intended. When the debugger finds an error, you correct the problem in your source code, recompile, and resume testing.

- **Testing**

  Before testing on actual devices, you can test on an emulator integrated with your development environment. By testing your MIDlets on multiple emulators, you can get an instant sense of how your application will look and function on the actual devices. Ultimately, of course, your wireless application should be tested on the actual device(s) it will run on.

- **Coding Errors**

  Three broad categories of errors can occur in your code: compile errors, runtime errors, and logic errors.

  - Compile errors appear at the time your source code is compiled. Syntax errors commonly appear at compile time.

- A runtime error causes the program to fail or terminate prematurely. Runtime errors result from problems that will not occur until the code is being executed. For example, trying to open a file that does not exist, or attempting to divide by zero, are errors that cannot be caught at compile time.

  In a typical development environment, when your program throws an exception, a debugger will print a stack trace in the Console output view, input view, and error view. You then double-click a line of the stack trace to go directly to the line of source code listed in the trace. As execution proceeds, you monitor the program's behavior step by step, until the error is produced or the program fails. At that point, you fix the bugs, recompile the program, and resume testing.
- A logic error is a non-fatal problem with the design or implementation of your application. Although the program statements may not throw exceptions, they do not produce the results you intended when you wrote the code. For example, logic errors can occur when variables contain incorrect values, when graphic images don't look right, or when output is simply incorrect. Memory leaks and attempting to insert more elements into an array than were allocated to the array can also result from logic errors.

  A debugger can help locate logic errors by allowing you to monitor the values of program variables and data objects as the program executes. Modifying data values during a debugging session is a good way to test hypothetical bug fixes.

	**Note:** Enabling optional compiler warnings in your IDE allows you to make full use of its debugging features. However, this can considerably slow down the process. In Eclipse, compiler warnings can be turned on or off by selecting in the Eclipse menu bar **Run > Debug > Nokia SDK Plug-in** and then choosing the appropriate warnings.

Common testing programs

The Java Verified testing program is an industry-wide initiative for testing J2ME™ applications. In the program, applications are tested against Unified Testing Criteria, which is developed and maintained by Unified Testing Initiative™ launched by Motorola, Nokia, Siemens, Sony Ericsson, and Sun Microsystems. It is recommendable to use Unified Testing Criteria (UTC) and also implement its requirements to the applications. For more information about Java Verified, see The Java Verified Program Web site at http://www.javaverified.com.

2.4.3 Basic considerations

This section lists the fundamental requirements of any MIDP application.

2.4.3.1 Content considerations

Application testing consists of more than technical testing. It is important that all general nontechnical issues are in condition as well. For example, the Intellectual Property Rights (IPR) must not be violated, and the form and content of the application must be generally acceptable, that is, not considered offensive by any parties.

Check that:

- Intellectual Property Rights (IPR) are in satisfactory condition.
- The content of the application is acceptable.
- If the application supports a technology that offers a chance for pirated content, it must also offer a technique for licensed content (for example, an MP3 player must support a DRM technique for MP3s).

2.4.3.2 Application performance and reliability

When creating applications, keep in mind that the speed of the application must not compromise the use and purpose of the application. This issue must be considered in the very early phases of application design. Depending on the type of application, remember that end users may use other applications concurrently. Therefore, excessive consumption of the device's running power and memory should be avoided.

Make sure that the application does not cause any harm to its user, the system's applications, or the data stored in the system. Harmful applications may be created accidentally, for example, if an application unnecessarily consumes a lot of the device's processing power or fills up the device's flash memory. Purposeless network communication must be avoided as well because it may create unnecessary expenses. Therefore, optimize the application by finding and correcting possible bottlenecks in the code, save only necessary data to the device's flash memory, and keep communication to outside the device reasonable.

Check that:

- The application speed does not compromise the use and purpose of the application.
- The application does not consume the device's running power and memory excessively.
- The application does not cause any harm to the user, other applications, or data.
- There are no bottlenecks in the code.
- Only necessary data is saved into the flash memory.
- Communication to and from the device/application is kept within reasonable limits.
- Occasional tasks, exceptional tasks (for example, for emergency conditions), and tasks that cope with errors (for example, caused by using the interruption of network connection during the application's use) must be considered and treated appropriately.

2.4.3.3 Application security

A mobile device is carried everywhere by its owners. Because of this fact, there is always a risk that the device will be lost. Therefore, no sensitive data should be stored in a mobile device without encryption. For example, the application must not save the credit card number to the device's memory. However, if the nature of the application requires saving sensitive data, the data must be encrypted and hidden behind a password. In addition, the user must be informed clearly that sensitive data is stored into the device's memory.

The application must not echo the input of sensitive data, for example, pins and passwords. However, the chosen character may appear briefly in order to confirm which character has been entered; however, the character must then be masked. This can be done, for example, by using the `TextField` class. Set the value of the input constraints parameter to `TextField.PASSWORD`, as shown in the following example:

```
TextField textField = new TextField("Password", "", 10,
TextField.PASSWORD);
```

Possible communication must be secure if sensitive information is transmitted between two end points or if a user may be charged for the result or consequences of a transaction. In such situations, use secure HTTP (HTTPS) or a secure socket connection. Both are supported from S60 2nd Edition onwards.

Encryption is also needed if sensitive data is transmitted via Bluetooth, Short Message Service (SMS), or Multimedia Message Service (MMS). The used Bluetooth stack provides security mechanisms and the Bluetooth implementation enables and disables security mechanisms. Encryption of SMS and MMS messages must be implemented by the developer.

The change in the security model for Mobile Information Device Profile (MIDP) 2.0 requires that user notifications are handled as the system has meant them to operate. For this reason those prompts cannot be interrupted.

Check that:

- Sensitive data is encrypted before it is stored into the device.
- Accessing Sensitive data requires a password.
- The user is informed when sensitive data is stored into the device's memory.
- HTTPS or secure socket is used for secure communication.
- SMS and MMS messaging as well as the Bluetooth connection are protected if needed.
- A MIDlet must not be able to simulate security prompts and notifications to the user generated by the system or virtual machine.

2.4.3.4 Installation

For a good user experience it is important that the application installation works smoothly and easily. Be sure that all mandatory attributes and their values in the Java Application Descriptor (JAD) file are typed correctly and that they are in the right format. The attributes and values of the Java Archive (JAR) package's manifest file must be correct as well. The best way to ensure this is to use a packaging tool, for example, Carbide.j. With this tool it is easy to set and update attributes and values into both the JAD and manifest files.

For information on JAR and JAD files, see section Default project structure.

Check that:

- All mandatory attributes and their values in the JAD file and the JAR package are correct.
- The application descriptor contains the attributes specified in the MIDP 1.0 and MIDP 2.0 specifications.
- Data reported in the JAD file is consistent with the information reported within the application.
- The maximum JAR file size does not exceed the limit for downloadable applications so that the application can be installed over-the-air, too.

2.4.4 Functionality testing

Make sure that all functions in the application work as they should. Test all possible extraordinary events and use cases during the application run – also, do things that you might assume users won't do.

2.4.4.1 Environmental changes

A mobile device can be used in a variety of environments and situations. Because of this, MIDlets must work without error if the environment changes rapidly. If, however, errors do occur, they must be handled gracefully.

Basically, it is extremely important that all exceptions are carefully handled in the code and the MIDlet run is guaranteed regardless of the exception. Make sure that the user is well informed in case of extraordinary situations. Also note that any error messages in the application must be clearly understandable. Error messages must clearly explain the nature of the problem and indicate what action needs to be taken.

Check that:

- MIDlets work correctly even though the environment may change.
- If errors occur, they are handled gracefully.
- The MIDlet runs even though exceptions are handled at the same time.

- The user is informed in extraordinary situations.
- Error messages are easy to understand by any end user.

2.4.4.2 Networking

An interruption of the network connection during the MIDlet run must also be considered and treated appropriately. Check that your MIDlet handles possible network errors and informs the user about them.

The application must also handle situations where connection is not allowed and close the connection after the session is over.

The system usually closes network connections when the application is about to close. However, it is recommended that the network connection be either disconnected by the user or programmatically before the application closes.

Check that:

- Network exceptions are handled and the user is notified about them.
- The network connection is closed when it is not needed anymore and/or the application is closing.

2.4.4.3 Wireless messaging

The Wireless Messaging API 2.0 (WMA) enables the sending of SMS and MMS messages (MMS was first implemented in S60 Third Edition).

Verify that the MIDlet successfully sends the SMS message. Make sure that the application gives the user an accurate and appropriate error message if the mobile device is unable to send the SMS message because of external factors (for example, network connectivity).

Check that the message is formatted appropriately if it is sent to the mobile device's mailbox. If the message is sent to an application, verify also that the SMS message is received and processed correctly by the receiving application.

Check that:

- The MIDlet sends SMS messages correctly.
- The user gets a clear error message if something goes wrong.
- The application sends and/or receives data packets properly.
- The message has an appropriate format and it is received and processed correctly by the application.

2.4.4.4 Bluetooth connectivity

The Bluetooth API provides access to RFCOMM, L2CAP, and SDP protocols.

Point-to-multipoint applications

Most Bluetooth applications use point-to-point connections between two devices. However, in certain applications, such as games, point-to-multipoint connections could provide added value. To get the best out of point-to-multipoint connections, a few conditions must be considered.

Note that not all Bluetooth devices support point-to-multipoint. Also, test your point-to-multipoint application with multiple devices to find out the actual number of how many devices can be used with your application. The Bluetooth specification defines seven as the maximum number of slaves, but in practice the device implementation or its resources may limit the number.

In current phones, scatternet is not widely supported. This means that a slave can only accept one connection at a time, and it cannot be a master while already being a slave in another piconet. This means that you can only establish multiple connections from one single device (master) to slaves.

In point-to-multipoint scenarios, the handling of interruptions becomes crucial. For more information on interruptions, see the guidelines in section Handling interruptions.

Check that:

- If you state that your application connects with, for example, seven devices via Bluetooth, make sure that it supports those seven devices and that the usability of the application does not suffer.

Optimizing device and service discoveries

Bluetooth device and service discoveries may take a while to complete. Therefore, it is advisable to optimize these procedures. In device discovery, use Limited Inquiry Access Code (LIAC) instead of General Inquiry Access Code (GIAC) whenever possible. With a LIAC inquiry, the application finds only the devices that are in Limited-Discoverable mode, thus speeding up the procedure. Using LIAC is especially feasible when discovering counterpart device(s) that are waiting for your connection, for example, to start a game. Please note that you have to set the counterpart device (application) into the Limited-Discoverable mode. In Bluetooth devices, Discoverable mode is usually set as the default.

You can also optimize the discovery procedures by stopping the discovery after the first match(es). In service discovery, for example, you can stop the search after you have found the first device(s) running the desired service of the application.

Check that:

- Device and service discoveries are efficient. Use Limited Inquiry Access Code to speed up the discovery procedure if feasible.

Power consumption considerations

In idle mode, Bluetooth does not significantly increase power consumption. However, when you have an active Bluetooth connection, it is worth considering disconnecting the link whenever no data is transferred for a long time to decrease power consumption. When the connection is reopened, use the `retrieveDevices()` method of the `DiscoveryAgent` to return an array of Bluetooth devices that have either been found by the local device during previous inquiry requests or have been specified as preknown devices depending on the argument of the method.

Check that:

- The connection is turned off if it has not been needed for a long time.

Handling interruptions

The application is responsible for possible recovery actions, for example, for reconnecting the link, for continuing to work after one device has left the session, and for notifying users about the situation. However, when implementing recovery features, keep in mind that the user interface must be logical in order to provide a good, intuitive user experience.

A typical interruption occurs when one of the devices receives an incoming call or a message. In this case, the Bluetooth application should locally go into the paused mode using the `hideNotify()` method as described in section Pausing a MIDlet. A Bluetooth application should also inform the connected devices about the pause (in the case of a point-to-multipoint scenario, the slave receiving a call or a message first informs the master, which in turn should then inform other slaves). The application has the liberty to use any proprietary message in order to command other devices into the

paused state, as there are no standardized methods. The Bluetooth link remains connected also during the paused state, so there should be no need to reconnect just for continuing the application.

There are also several cases that may happen to a remote slave device: It may move out of range, run out of power and eventually be switched off, or its application may be closed (disconnected) by the user. For the master-side application, all of these cases look as if a remote device was disconnected. The slaves cannot see if one of them is disconnected. Therefore it is the master-side application's responsibility to inform the remaining slaves about the changed situation. There are several possible ways to handle these cases. The application could be continued by the remaining parties or the master could attempt to reconnect to the disappeared device. In the latter case, it is possible that the device won't be found again because, for example, its application may not have been restarted.

It is a trickier scenario if the master disconnects from the application, because it leads to the dismantling of the whole piconet (the slaves cannot run the piconet without a master). There is the possibility for any slave to start discovering the devices and become a new master. The game should be saved so that it can be continued. It is up to the developer to evaluate if this kind of recovery is feasible and reasonable to implement.

The system usually closes connections when the application is about to close. For convenience, it is recommended that the Bluetooth connection is either disconnected by the user or programmatically before the application closes.

Check that:

- Other devices continue the Bluetooth session even though one device would disappear from the session.
- The user is informed in case of interruptions.

2.4.4.5 Personal information management

The PIM API is an optional J2ME package that gives support for accessing and modifying the PIM databases that may exist in a MIDP device. The PIM API currently supports three types of databases or lists: Contact lists, Event lists, and To-Do lists. In MIDP 2.0 devices, the user data can be manipulated with two functions groups: "Read User Data Access" and "Write User Data Access" groups. If the PIM database lists are being used, the function groups mentioned above must also be used.

Check that:

- If reading and writing user data is not allowed in the settings, the application must show an informative error message.
- The error message is triggered automatically from the PIM API or directly from the application, saying that the reading or writing was not possible.

In case user allows access to PIM applications, there a certain things that must be considered when handling user data. First of all, user data must be handled correctly and secondly, the user data cannot be modified or destroyed without asking the user's permission.

Check that:

- User data is read and written correctly.
- User data is not modified or destroyed without the user's permission.

2.4.4.6 Record Management System (RMS)

As stated before, only necessary data should be saved into the device's flash memory. Optimize the data, if possible, and use packing algorithms if a lot of data must be saved. However, compare the amount of extra code for packing algorithms with the benefit of the size of packed data.

Remember to inform the user if overwriting a significant entry in the database is irreversible.

It is important to handle exceptions and be prepared for environmental changes when saving data to the flash memory of the device through the Record Management System (RMS). S60 devices do not have a strict limit for a MIDlet's RMS space, and basically MIDlets can save as much data to the flash memory as there is free memory available. However, remember that another application (Symbian or Java) may have consumed all the flash memory – in that case the system throws a `RecordStoreFullException` error message.

It is also possible that an extraordinary environment change may take place while saving or reading data to/from the RMS. For example, if the device's battery gets empty while saving data, it is possible that the data will not be saved into the flash memory or the saved data will be corrupted. Make sure that the application works without errors even if the saved data includes old information or is broken.

Note that if an application is updated, the RMS data of the older version may remain in the flash memory of the device. Check that the existing data is compatible with the newer version or that it is deleted programmatically. Note that when installing a newer version, the user might also be asked whether to delete all data related to the older version.

Check that:

- The device's flash memory is not filled with unnecessary data.
- The user is informed if important data is about to be overwritten in the database.
- The application is prepared for exceptions and errors.

2.4.4.7 Pausing a MIDlet

In many cases it is useful or even necessary to have the MIDlet pause itself automatically. For example, all possible system notifications are shown over the MIDlet, thus the MIDlet is hidden in the background. Note that with Series 40 devices, a phone call, for example, interrupting the application is the only way for the application to enter the Paused state.

MIDP specifications define that a system can set MIDlets to the paused state (the `pauseApp()` method) if the system needs a lot of the device's resources. In the paused state, the MIDlet frees all its resources. The currently available Series 60 devices have sufficient running power and memory, and they do not need to enter the Paused state. This means that you must create your own pausing mechanisms.

In other words, you must create a method that pauses an active session when one of the following events occurs during the MIDlet interaction:

- A system notification is shown on the screen (for example, an incoming call, or notification of battery full or low).
- The user presses the red dial key, the power key or the application key of the device.
- The main screen of the application is hidden by a system menu, or another application is set to run on the foreground.

In practice, the MIDlet must always be paused when it is hidden. This is especially important in game applications, because the player will probably lose the game if it is not immediately paused when the game is hidden.

MIDlets can be paused with the `isShown()` method of the `Displayable` class or the `hideNotify()` method of the `Canvas` class.

All User Interface (UI) components are inherited from the `Displayable` class. Thus, the `isShown()` method can be used to test whether a UI component is visible or not. Repetitive requesting of the `isShown()` method can be used to get information if the UI component is hidden for some reason. The `isShown()` method can be used for

high-level UI components like `Form` or `List`, but in case of low-level UI components that are inherited from `Canvas` class it is better to use its proprietary `hideNotify()` method.

The `hideNotify()` method is called right after the `Canvas` object has left the display. Create an auto-pause mechanism inside the `hideNotify()` method to stop threads, cancel timers, save important values, and so on.

After the paused session is ready to be continued, the Continue option needs to be shown to the user. It can be, for example, one entry of the main menu of the application.

Check that:

- The application can be paused when any interruption happens.
- The MIDlet pauses itself automatically if it is hidden.
- It is possible to choose to continue the application after the interruption.

2.4.5 Usability testing

This section lists only the basic requirements for usability testing.

Good usability is the basis for a good application and a pleasant user experience. If the application's usability is poor, some users might stop using it. To ensure user satisfaction, you should arrange separate usability tests and start testing the application already in the early phases of application development. It is easier to fix possible usability problems in these earlier stages.

For a positive user experience, it is important that all main functionalities of the application can be accessed easily through the main menu. Ensure that each functionality works properly as described in the application documentation and the Instructions section and that there are no hidden functionalities.

Check also that the terminology is correct and that there are no grammatical mistakes. Do not use difficult terminology or abbreviations.

Make sure that the application gives enough feedback to the user. Typically users do not wait for more than a couple of seconds if the display does not change and seems jammed after they have selected a function. In addition, if the application repeatedly shows certain notifications or questions to the user, it must be possible to turn those off.

Any selection of a different function in the application should be performed within five seconds. There must be a visual indication within one second that the function will be performed. The visual indication can be, for example, prompting for user input, using splash screens or progress bars, or displaying text such as "Please wait..." or "Sending message…". These indications can be enhanced with sounds. Consideration is possible for applications that are subject to Digital Rights Management or other types of verification in the start up. In these cases, the waiting time can be longer.

Check that:

- All main functions of the application can easily be accessed through the main menu.
- Each functionality works as described in the documentation and in the application's Instructions section.
- Terminology and grammar are correct and no abbreviations are used.
- Each screen appears for the length of time that is necessary to read its information.
- The consistency of the following features is maintained throughout the application: terms, layout, colors (or reverted colors), softkey labels, vibra, and sounds.
- The application gives enough feedback to the user.
- Repeated user prompts can be turned off.

Menus

Check that menu structures are grouped logically and not too deep. A wide menu hierarchy is preferred over a deep one. The user must not get lost in the menus. Ensure that the Back button is always at hand, on the right softkey. Closing the application must be easy as well.

The order of the items in the menus must be well defined. It is advisable to arrange items by frequency of use, that is, the most frequently used item should be at the top of the menu, and the Exit item should be the last item, at the bottom. In S60 devices, until S60 Platform 2nd Edition Feature Pack 3, the Exit item is automatically on the Options list of the left softkey.

All menu items must be shown as selectable items, and the labels of the items must be descriptive. The user must easily see which screen items can be selected and/or changed, especially if low-level UI components are used as menus. Selectivity is not a problem if high-level UI menu components are used because they are similar to the system menus.

If the application requires completing multiphased forms, the current screen number as well as the total number of the screens must be indicated to the user in, for example, the following format: Page 3/5. The last screen in the form must give the user some informative feedback, for example, a Thank You screen after pressing the Submit button.

Add Help to the main menu of the application. Help must be useful and complete and include at least the purpose of the application, rules, and use of the keys.

The application also needs to contain an About screen that gives information about the developer and other generic information about the application. The About screen should contain developer name, application name and application version number. These values should be the same as defined in the JAD file.

Check that:

- Sequences of actions are grouped logically.
- Menus are grouped logically and menu structures are not too deep.
- The Back or Exit button is always available on the right softkey.
- The order of the menu is logical.
- Menu items are marked clearly.
- The application has a Help section.
- Information about the developer and the application is given in an About section.
- It is possible to Exit the application from the main menu.

Commands

The order of commands and the softkey to which they are assigned depend on the following:

- Command type
- Command priority
- The order in which the commands were added to the UI component

For more information about command mappings and command labels in S60 devices, see S60 Developer Platform 2.0: Specification .

In S60 devices, all UI components, except for the Nokia UI API's `FullCanvas` class, include an Exit or Close command by default. Be careful that UI components do not end up with two Exit commands, for example if you have added one yourself.

Check that:

- Softkey labels reflect the style of the user interface of the particular Developer Platform.
- The label related to the right softkey refers to Back, Quit, Exit, Cancel, Clear, or other “negative/backward functions.”

Sounds

Ensure that each sound (if sounds are used) in the application is distinctive and has a well-defined meaning. For example, a happy sound should be used for a positive action and a sad sound for a negative action.

Be sure that the current status of the possible sound settings does not affect the playability of the application (for example, whether or not the sound is activated).

All in all, you should be careful when adding sounds because too much sound may make the use of the application annoying. Therefore, add sounds only to places and situations where they give more information or an extra experience for the user. Be sure that sounds can be turned off easily.

Check that:

- Each sound has a distinctive meaning.
- Sound settings do not affect the playability of the application.
- Sounds are not overused in the application.
- Sounds can be set on/off in the application and setting is saved on exit.

Backlights and vibra

The backlights and vibra functionalities are supported via the MIDP 2.0 APIs.

The vibra functionality can be used to give extra feedback to the user. Note that it is a wearing functionality, so it should be used only in special circumstances and less frequently than sounds. There should an easy way to set vibra on/off and the setting is saved on exit.

Backlights should always be on if the application is actively used without user interaction (there have been no key presses to keep the backlights automatically on). However, backlights consume quite a lot of battery power, so there should be a setting that allows them to be turned off.

Check that:

- Vibra is not used too often.
- Vibra can be set on/off.
- The user can turn backlights off.

Settings

It is advisable to have user settings for the application, for example, settings for sounds. Regardless of the settings it is important that all settings and selections can be canceled or restored. Also, make sure that the current status of each setting has a clear meaning and that the user can easily understand what changes will occur if a setting is changed. Note that settings must be saved when the application is closed.

Check that:

- The user can cancel and/or restore settings and selections.
- The current status of each setting is clear.
- Settings are saved when the application is closed.

Localization

The content in all different views in the application must be available in the application's target language. It means that e.g. Main menu, Help and About are localized. Text should not contain any spelling errors and it should be clear and readable (e.g. no overlapping or text cut off).

The date format must be handled according to the target country. Verify that date, time, time zone, week start, numeric separators, and currency are formatted according to the conventions of the target country of the implemented language and supported throughout the application.

Data entry fields must accept and properly display national alphabets, for example, ä, ü, and ß. If possible, test that all data entry fields accept and properly display all special character input.

Check that:

- All text is available in the application's target language.
- There are no spelling errors and text is not overlapped or cut off.
- The language conventions of the target country are followed.
- National alphabets are accepted and displayed.

Games

The following guidelines are especially concerned with game MIDlets. It is important that the general view of the game is clear and that the game gives enough feedback and immediately responds to the player.

The rule or function of each element in the game must be clear (for example, how the player and the enemies are represented, and how they interact).

The game must be paused automatically as described earlier (section Pausing a MIDlet) and it is also important that the player can pause the game by him/herself. When the game is paused it must be easily continued.

If the application is closed in the middle of the game, the game status and settings must be saved so that the game can be continued later.

Check that:

- The general view and the rules and functions of the game are clear and described in the Instructions.
- The game gives enough feedback to the player.
- The user can pause and continue the game and the status is saved.
- The current status of the settings does not affect the playability of the game (for example, whether or not the sound is on).

2.4.6 Typical errors in Java ME applications

This section lists the most typical errors that developers have encountered over the years.

Pay particular attention to these issues and make sure that they have been resolved in your application.

- The application does not pause during interrupting operations, for example, an incoming phone call. (This is not necessary for all kinds of applications.)
- The application cannot be downloaded over the air/installation does not work.
- The Instructions section is not complete; it does not include the purpose of the application, use of keys, or how to use the application and its functions.

- Proceeding from the startup or other screen takes more than five seconds and the user receives no guidance about what to do next.
- The application includes grammatical and spelling mistakes.
- Settings are not saved when the application is closed.
- Error messages are incomplete or do not provide sufficient information to the user.
- Application errors due to lack of memory or exceptions are not correctly handled.
- The application's dialogs, interaction, controls, softkey labels, UI layout, and so on, are inconsistent.
- The application’s name and version number are inconsistent or faulty.
- Rapid pushing of certain buttons creates difficulties in the application leading to crashes or jams.

2.5 Deploying MIDlets to a device

Once the MIDlet has been run on an emulator, the next step is to run it on the target device. There are a number of ways to deploy a MIDlet to a device. The method chosen will depend mainly on the hardware available. The main deployment methods are: Over-the-Air (OTA), Infrared, installing over a serial cable, Bluetooth, e-mail, and MMS. This section also provides a step-by-step guide to deploying a MIDlet using OTA provisioning.

One possible deployment option is to install and integrate the Carbide.j tool and use it for the deployment. For more information about this option, see the Carbide.j User’s Guide .

This section presents the following deployment methods:

- Over-the-Air (OTA)
- Infrared (IR) and Bluetooth technology
- Serial Cable
- E-mail and MMS

2.5.1 Over-the-Air (OTA)

With OTA deployment, the MIDlet is installed on a Web server and then downloaded to a device via the device's Internet microbrowser. A JAR file and a JAD file are required for OTA installation.

The steps involved in a typical MIDlet suite installation are shown below.

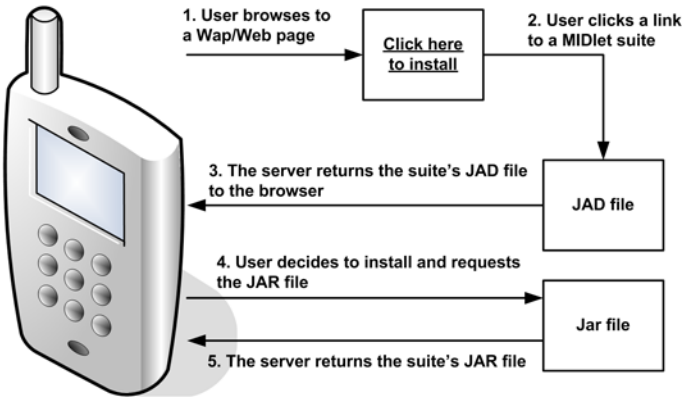


Figure 4: Typical MIDlet installation process using OTA

In the figure, the process begins by browsing to a page that includes a link to a MIDlet suite. The link is then clicked, which causes the Web server to deliver the JAD file for the MIDlet suite to the browser.

On receipt of the JAD file, it is opened and its contents are displayed. The JAD file contains a description of the MIDlet suite, including its size, version number, and who provided it. A request is then sent for the JAR file and, finally, the Web server delivers the JAR file. Once the JAR file has been delivered to the device, the suite is installed and individual MIDlets can be selected and run.

To deploy a MIDlet on a device using OTA provisioning:

1. Edit the JAD file. Change the `MIDlet-Jar-URL` attribute so that it points to the JAR file using an absolute URL. For example, if the Web server is running on the local machine, the URL will look something like: `http://localhost:8080/HelloWorld.jar`. For more information JAR and JAD files, see section Default project structure.
2. Install the JAD file and JAR file on the Web server.
3. Create a Wap or Web page with a hypertext link to the JAD file. The hypertext link must also be an absolute URL. For example:

```
<wml>
<body>
<a href=http://localhost:8080/HelloWorld.jad>Click here to install</a>
</body>
</wml>
```

4. Configure the Web server's MIME types so that JAD files are returned with MIME type `text/vnd.sun.j2me.app-descriptor` and JAR files are returned with MIME type `application/java-archive`.
5. Point the device's Wap browser (if a Wap page has been created) or Web browser (if a Web page has been created) to the URL of the created page. Click on the link.
6. Every device that supports the Java platform will have a piece of software for application delivery and management. As this software is device specific, the term Application Management Software (AMS) is generally used to describe it. Once a link to a JAD file has been clicked, the device's AMS will activate and display the name, version, and origin of the MIDlet suite.
7. With the details of the suite displayed, the AMS should give an option to install the suite.
8. The AMS should now also provide the ability to run any of the installed MIDlets. Run the MIDlet that has just been installed.

2.5.2 Infrared (IR) and Bluetooth technology

IR or Bluetooth technology can be used to send the MIDlet from a PC to another device. (They can also be used to send a MIDlet from one device to another, but this document is about MIDlet development and development will be performed on a PC). These methods of deployment are dependent upon having access to an IR pod or, if a laptop is used, on the laptop having an IR port or Bluetooth connection. How the device handles the sent MIDlet will be device specific. For example, the MIDlet may go into the Messaging Inbox, the AMS may be automatically or manually activated and handle the MIDlet, etc.

2.5.3 Serial Cable

If the device has a serial cable port and connectivity software for a PC, the MIDlet can be installed on the device over a serial cable. Again, how the received MIDlet is handled when it reaches the device will be device specific.

2.5.4 E-mail and MMS

A JAR file could be included as an attachment to an e-mail or within an MMS. When the e-mail/MMS is opened, the AMS will automatically be activated and provide an option to install the sent MIDlet.

2.6 Example: Creating an application with Eclipse/Carbide.j

2.6.1 Introduction

This section will take you through the main steps of creating a simple MIDP application with Carbide.j (integrated with Eclipse) and then running it on the S60 emulator. Instead of creating the code from scratch, the HelloWorldPlus example MIDlet will be used to demonstrate the main steps of the application development process.

The main steps are:

- Setting up the development environment - Installing and configuring the needed software.
- Creating a new MIDP project in Eclipse - Creating a "MyHelloWorld" MIDP Eclipse project for developing the application.
- Importing S60 example application source files to an Eclipse project - As the new MIDP project does not contain any source files, these will be imported from the HelloWorldPlus example application.
- Editing the source files - Creating content for the "MyHelloWorld" world is done by editing the apporpriate source files of the imported HelloWorldPlus example application.
- Creating the MyHelloWorld MIDlet - Creating the MyHelloWorld MIDlet with Carbide.j.
- Running the MyHelloWorld MIDlet on the S60 emulator - Running the MyHelloWorld MIDlet on the S60 emulator and viewing it there.

2.6.2 Setting up the development environment

Before creating an application with Carbide.j , you need to set up the necessary development environment. This means installing and configuring the following software:

1 Java™ J2SE Platform. Please refer to java.sun.com/downloads/index.html at http://java.sun.com/downloads/index.html for instructions on how to do this.

2 S60 3rd Edition SDK for Symbian OS Supporting Feature Pack 1, for MIDP

Please refer to the *SDK Installation Guide* for detailed instructions and SDK installation requirements.

3 Eclipse 3.1.2 (or newer)

Please refer to www.eclipse.org at http://www.eclipse.org/ for information on how to download and install Eclipse.

4 Carbide.j

Notice, that you need to integrate Carbide.j with Eclipse when installing it. Please refer to Installing Carbide.j for more information.

You also need to configure Carbide.j with the SDK after installation. Please refer to for more information.

Note: Once you have installed Eclipse and Carbide.j, you still need to configure the following before starting application development:

- Configure Carbide.j with the SDK (see Configuring SDK emulator for Carbide.j).
- Set Java debug preferences for Eclipse (see Setting Java debug preferences for Eclipse).

2.6.3 Creating a new MIDP project in Eclipse

Context

The stand-alone version of Carbide.j does not maintain any project-related information as such. Thus the instructions here describe how to create a new project with the Eclipse IDE with which Carbide.j has been integrated, as instructed in Installing Carbide.j. (For information on using Carbide.j as a standalone application, please refer to the Carbide.j *Users Guide*, which you can access through the Windows Start menu: **Start > All Programs > Carbide > Carbide.j 1.5 > Users Guide**.

Steps

1. Open Eclipse
2. Select **File > New > Project** from the menu bar.

   The **Select a wizard** view is displayed.

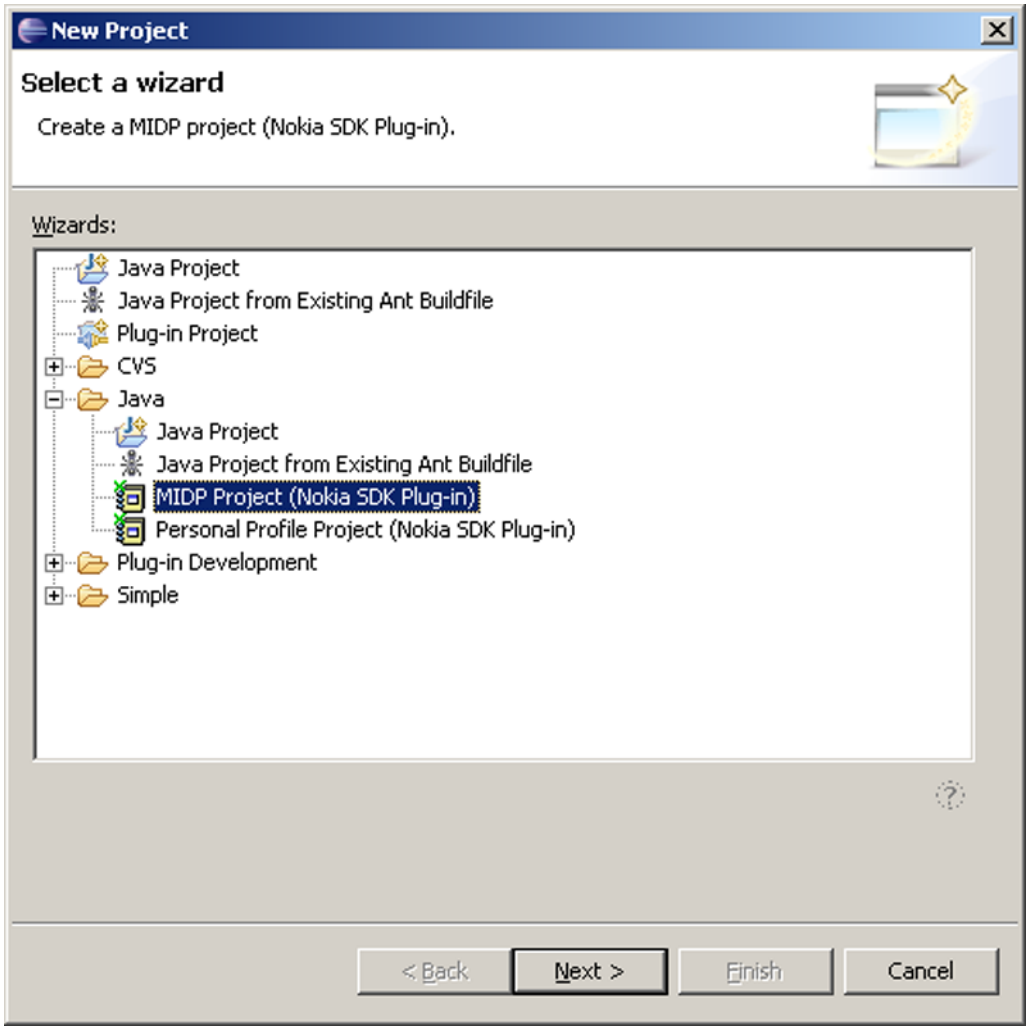


Figure 5: Select a MIDP project wizard

3. In the **Select a wizard** view, select **MIDP Project (Nokia SDK plug-in)** under **Java** and click **Next**.

   The **MIDP Project** view is displayed:

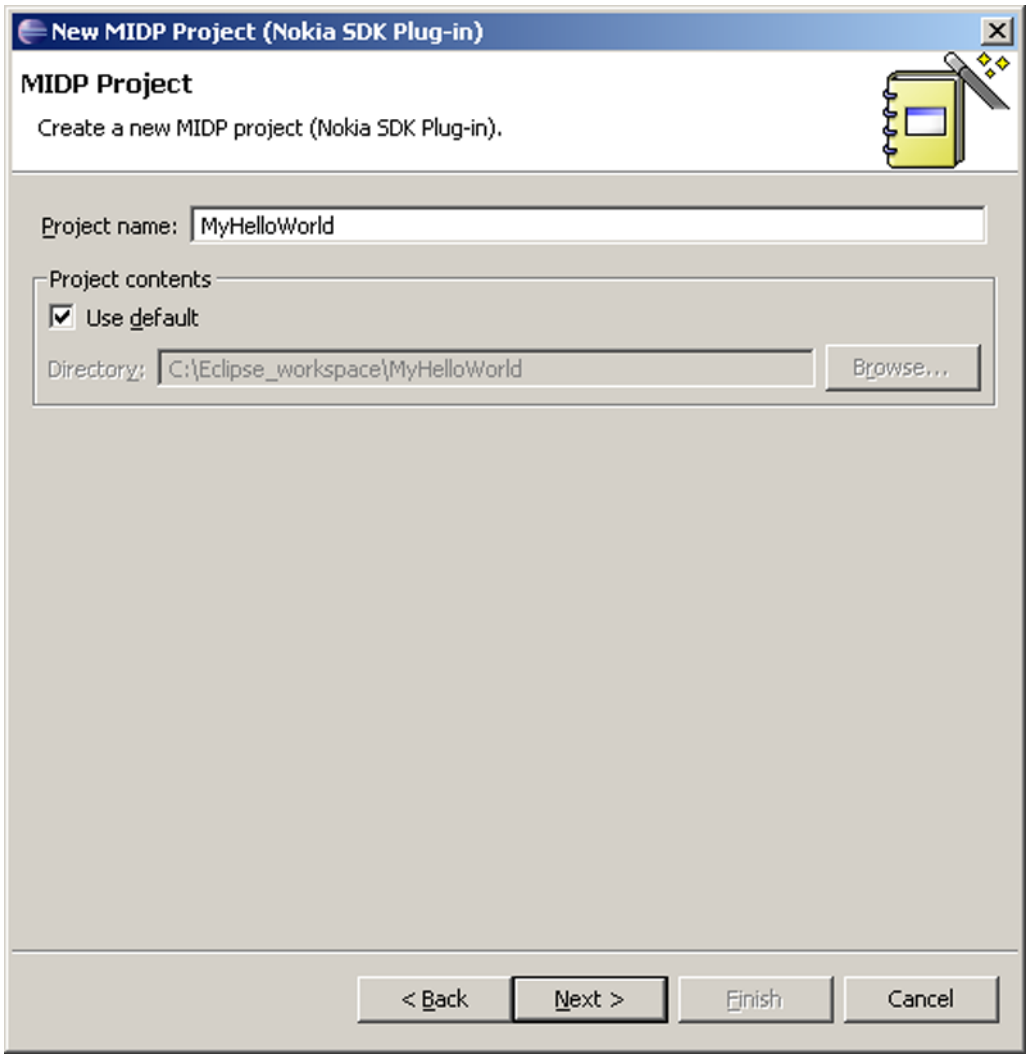


Figure 6: MIDP Project view

4. In the **MIDP Project** view, enter a name for your project (for example, "MyHelloWorld") and click **Next**.

   The **SDK selection** wizard view is displayed:

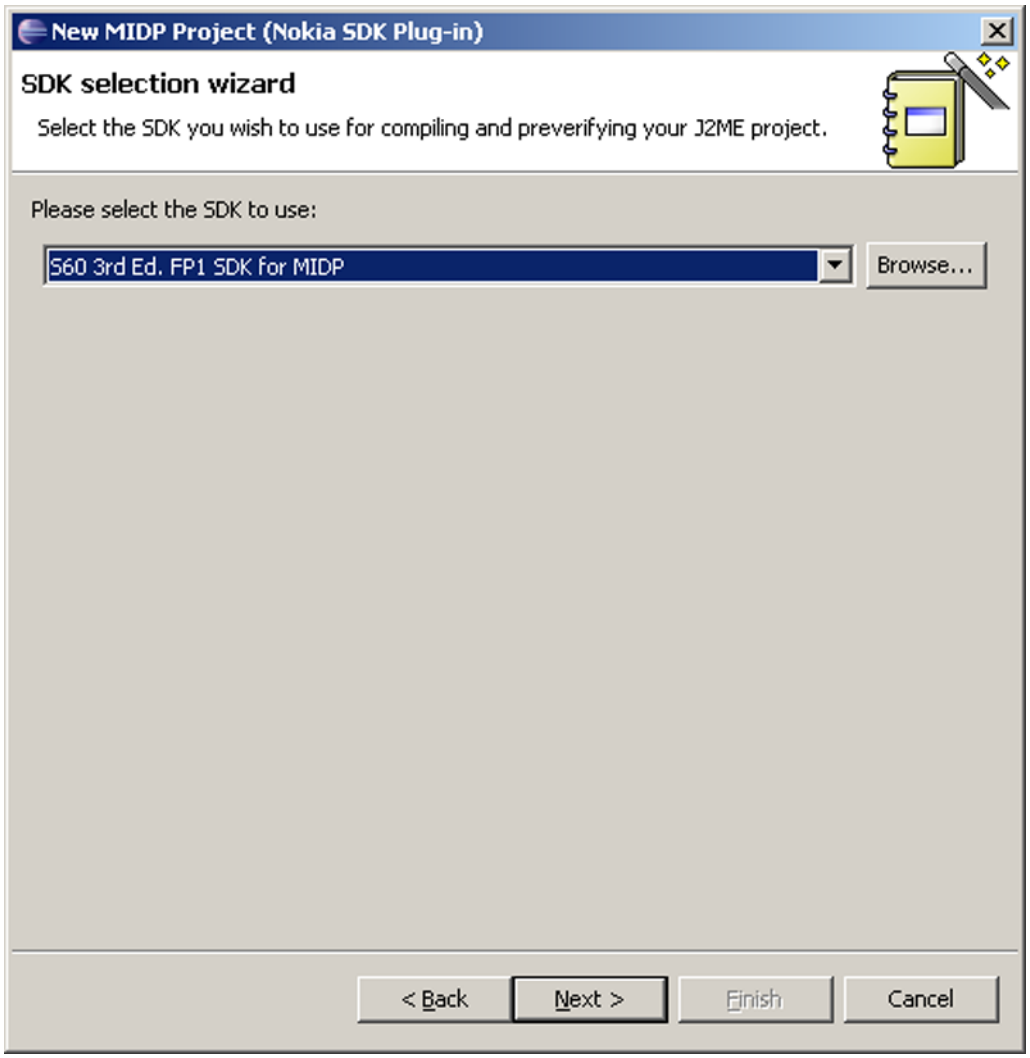


Figure 7: SDK selection wizard

5. In the **SDK selection wizard** view, select **S60 3rd Ed. FP1 for MIDP** as the SDK to be used and click **Next**.

   Note: In order to be able to select **S60 3rd Ed. FP1 for MIDP** as the SDK to be used, you first need to configure this SDK for Carbide.j. Please refer to Configuring SDK emulator for Carbide.j for more information on how to do this.

   The **Designer selection wizard** view is displayed:

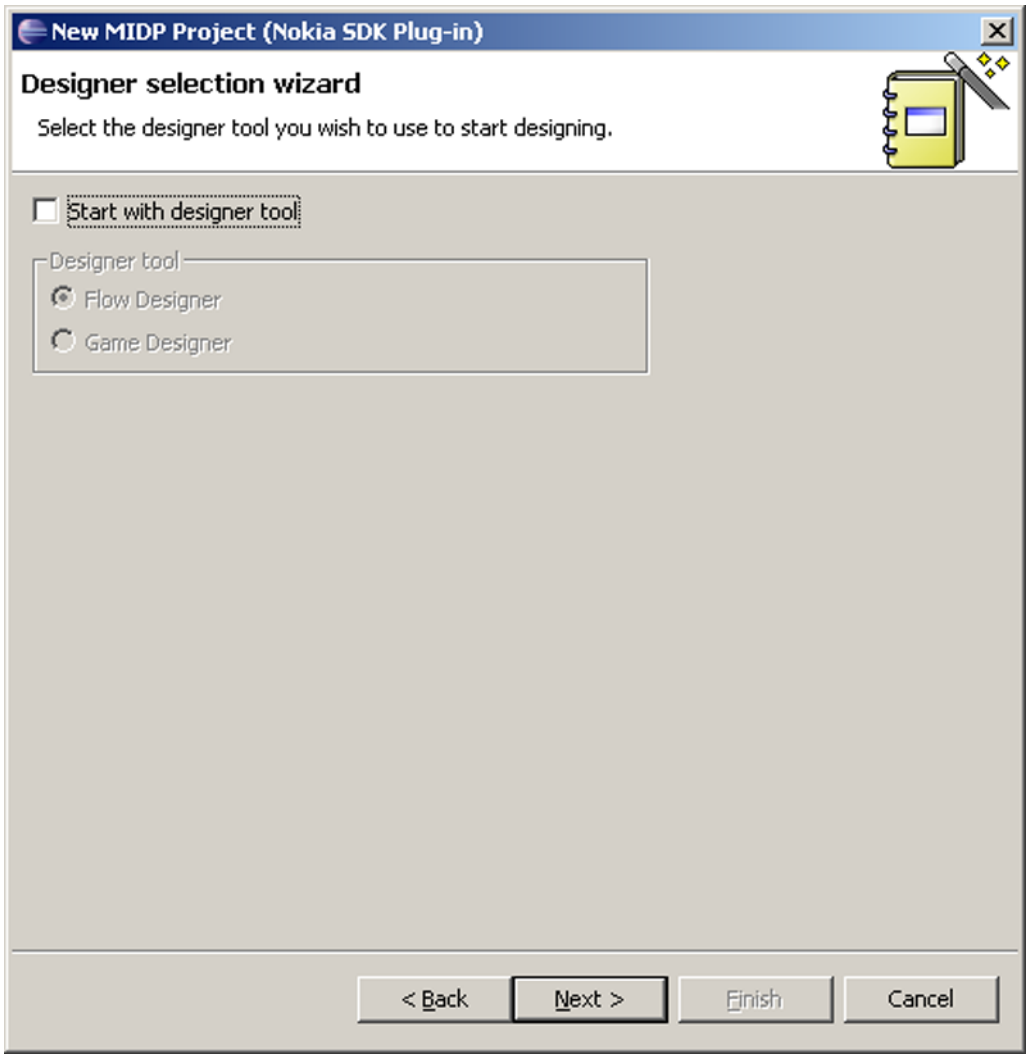


Figure 8: Designer selection wizard

6. In the **Designer selection wizard** view, uncheck the **Start with designer tool** checkbox and click **Next**.

   The **MIDP settings** view is displayed:

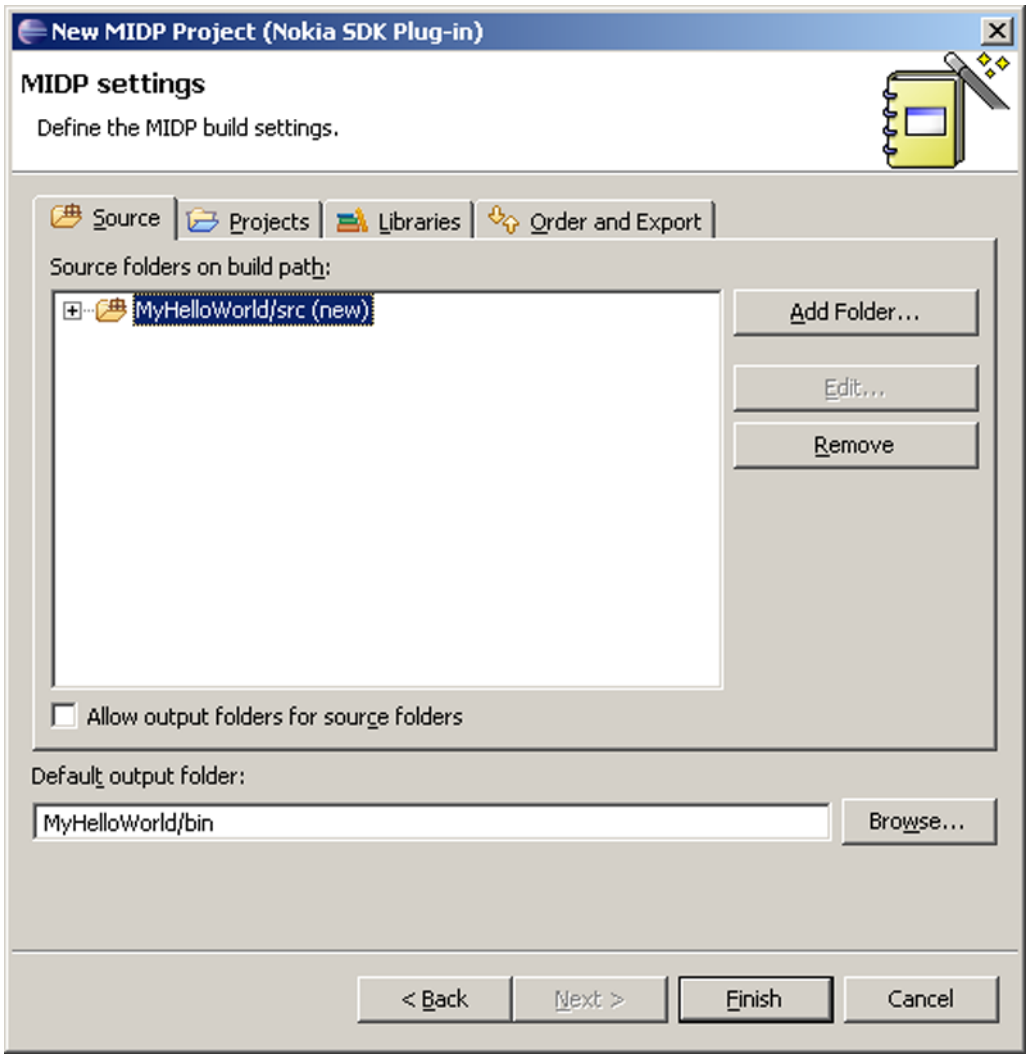


Figure 9: MIDP settings view

7 In the MIDP settings view, click **Finish**.

.

Results

You have now created a new "MyHelloWorld" MIDP project in Eclipse. As such, the project does not contain any source code. The Importing S60 example application source files to an Eclipse project topic instructs you how to import source code to your project from an S60 example application.

2.6.4 Importing S60 example application source files to an Eclipse project

Context

You can use the S60 example MIDlets delivered with the SDK to develop your own applications by, for example, importing source files from these to an Eclipse project.

Steps

1 Click the **src** folder of the project in the **Navigator** pane.

.

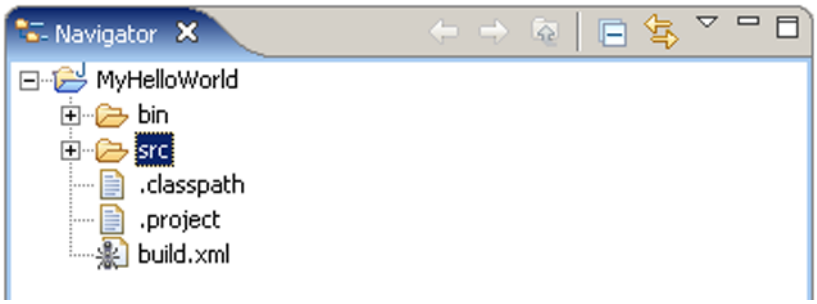


Figure 10: src folder in the Navigator pane

2 Select **File > import…** from the menu bar

. The **Import** dialog is displayed:

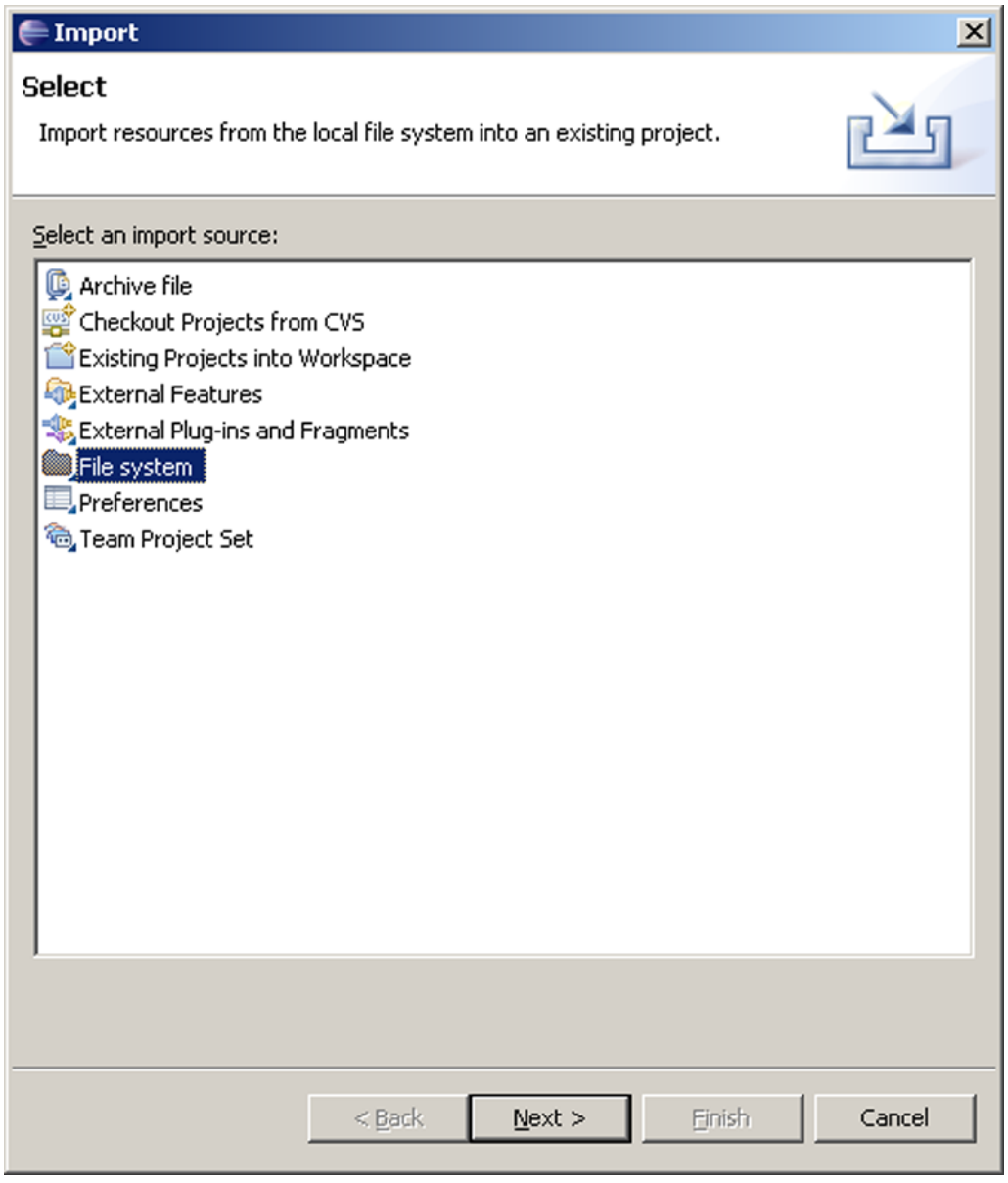


Figure 11: Import dialog

3 In the **Import** dialog select **File system** and click **Next**.

- The **Import File system** dialog is displayed:

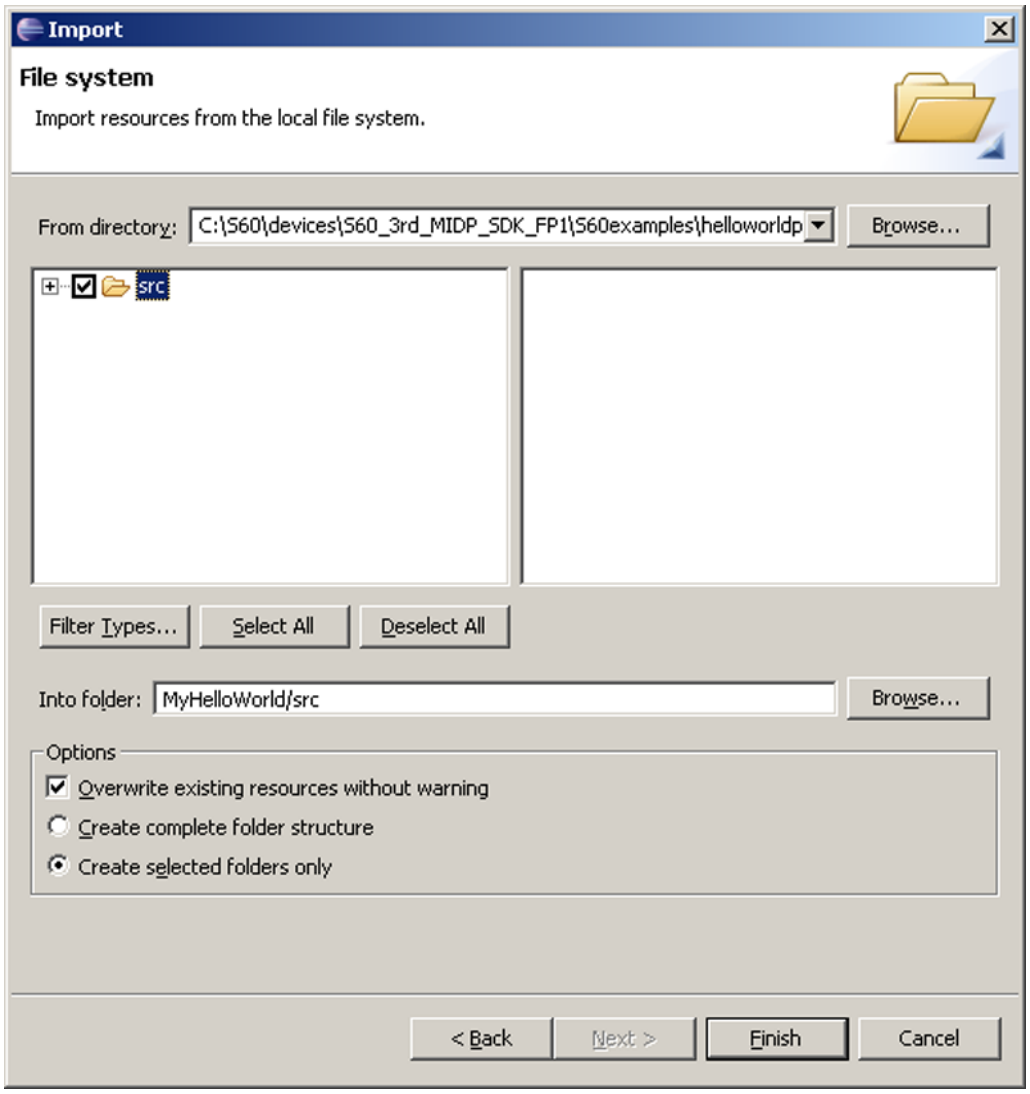


Figure 12: Importing File system

4 In the **Import File system** dialog,

- i Browse to the `\src` folder of the S60 example application whose source files you want to import and click **OK**.
  ii In the **Import File system** dialog, click the **src** checkbox.
  iii Click **Finish**.

Results

The source files of the HelloWorldPlus example application are imported to Eclipse:

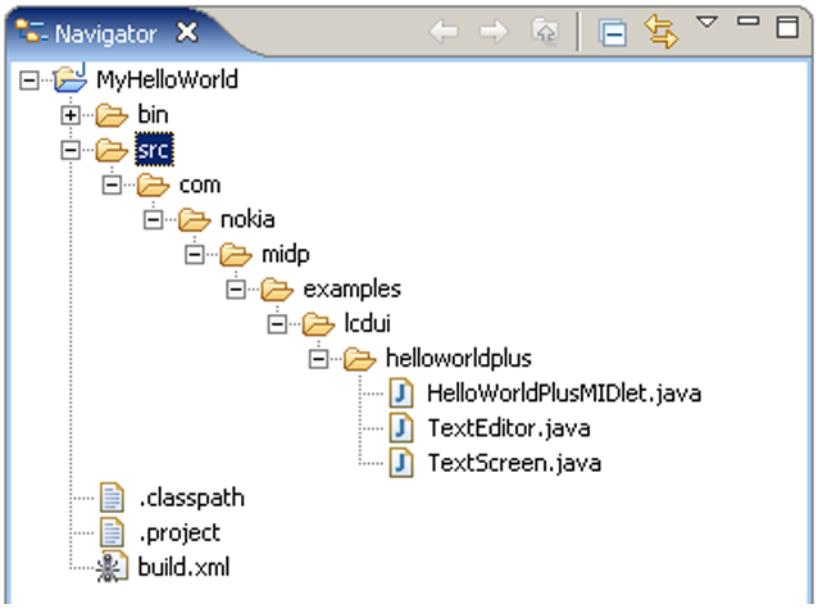


Figure 13: HelloWorldPlus source files

You can use these source files to create an application. See Editing the source files for an example on how to create a personalized HelloWorldPlus application by editing these files.

2.6.5 Editing the source files

Context

In this section we will create a MyHelloWorld MIDP application by editing imported source files of the HelloWorldPlus S60 example MIDlet.

The HelloWorldPlus example application displays a simple "Hello World!" message on the screen of an S60 device. In the MyHelloWorld application (see Creating a new MIDP application in Eclipse) we will change the message by editing the `HelloWorldPlusMIDlet.java` source file.

Steps

1 In the Eclipse Navigator pane, double-click `HelloWorldPlusMIDlet.java` .

. The file is opened in the Eclipse editor window:

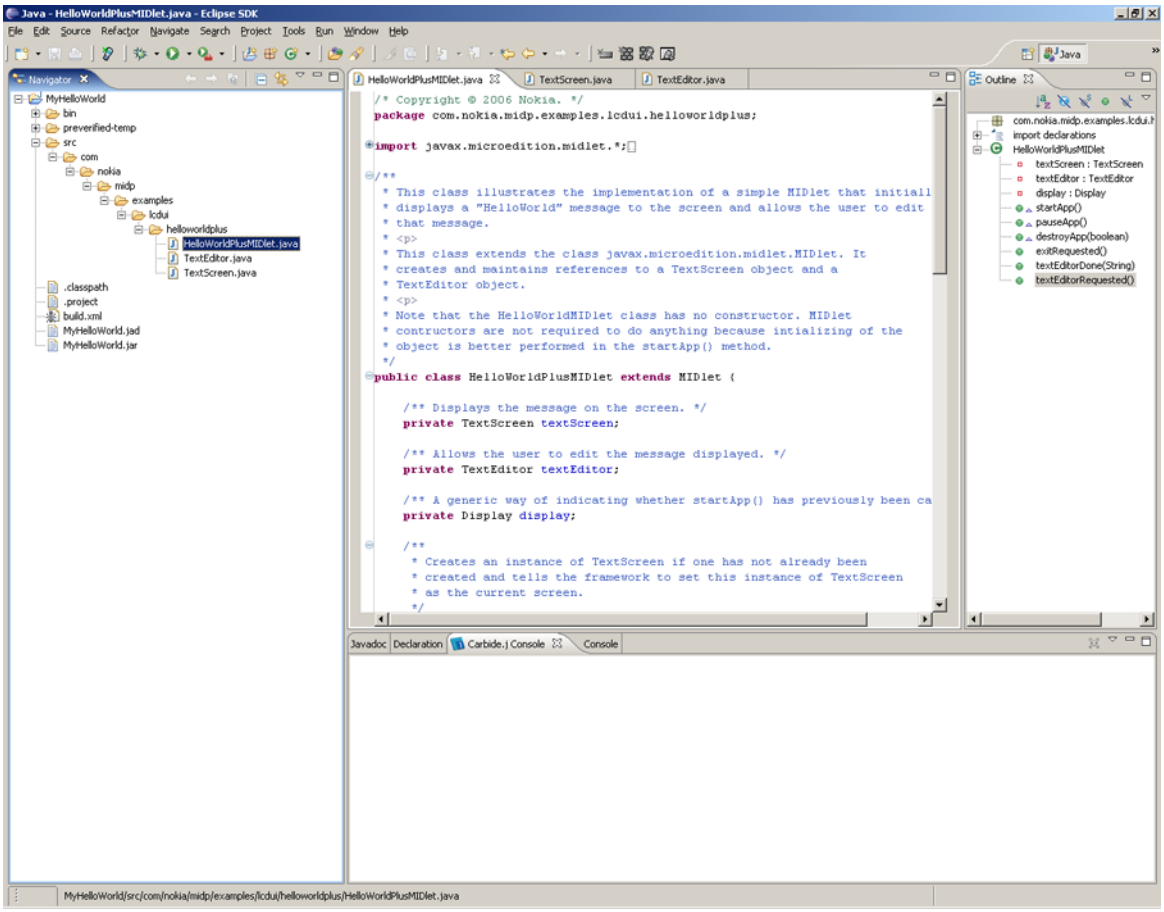


Figure 14: HelloWorldPlusMIDlet.java in editor window

2. Locate the "Hello World!" string in the `HelloWorldPlusMIDlet.java` source file.

   The "Hello World!" message that we want to edit is located under the `startApp()` method defined in the `HelloWorldPlusMIDlet.java` source file:

```
public void startApp() {
        if (display == null) {
            // First time we've been called.
            display=Display.getDisplay(this);
            textScreen = new TextScreen(this, "Hello World!");
        }
```

3. Change the "Hello World!" string to "Just Java it!" by editing the relevant section in the `HelloWorldPlusMIDlet.java` :

```
public void startApp() {
        if (display == null) {
            // First time we've been called.
            display=Display.getDisplay(this);
            textScreen = new TextScreen(this, "Just Java it!");
        }
```

4. Select **File > Save** from the menu bar to save your changes.

Results

The message displayed in the MyHelloWorld application has now been changed to "Just Java it!".

In order to be able to view and test the application in the S60 SDK emulator you need to first create a MyHelloWorld MIDlet from the project source files and then run this MIDlet on the emulator.

2.6.6 Creating the MyHelloWorld MIDlet

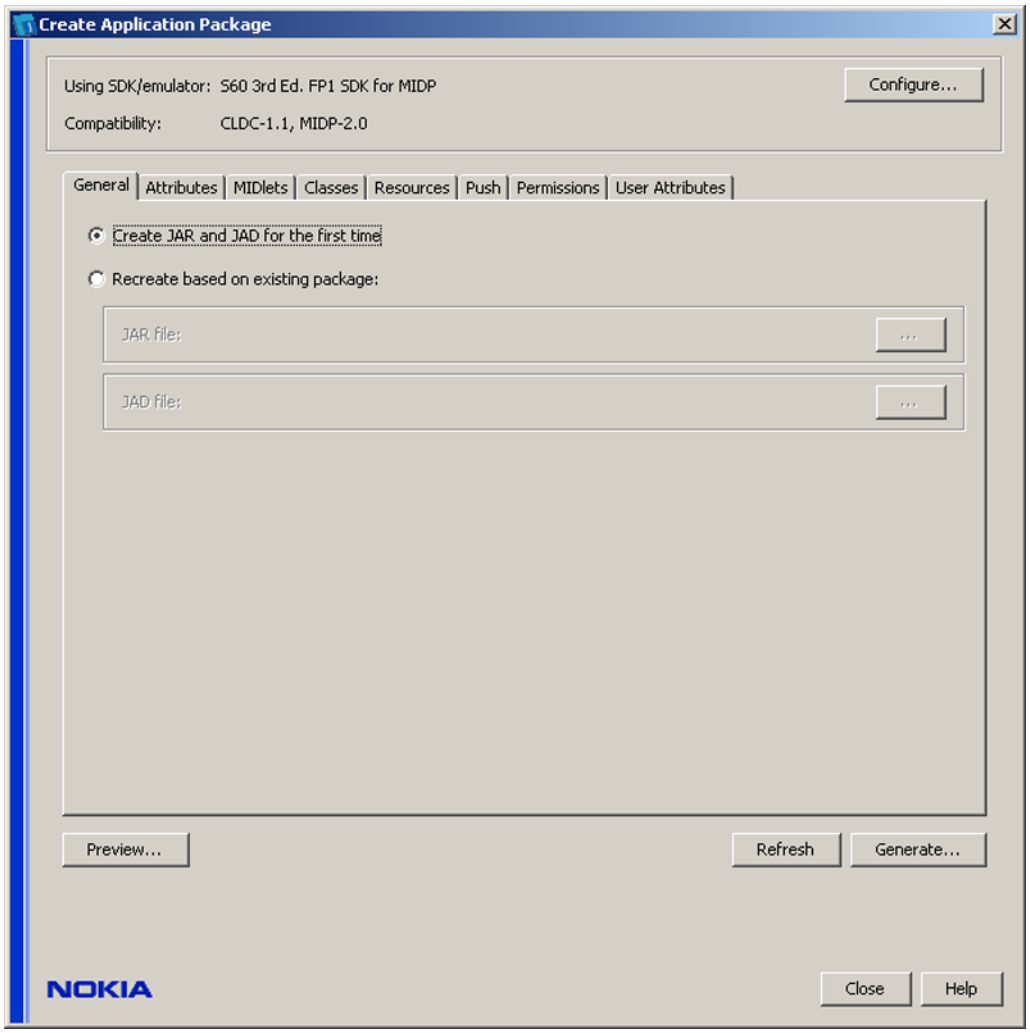
Context

In order to be able to run your MyHelloWorld application on the S60 emulator, you need to create a MIDlet out of it.

Steps

1. With the MyHelloWorld project open in Eclipse, select **Tools > Carbide.j > New Application Package...** from the menu bar.

   The **Create Application Package** dialog opens:



2. In the **Create Application Package** dialog, enter (at least) the following information:
   i. In the **General** tab, select **Create JAR and JAD for the first time**.
   ii. In the **Attributes** tab, enter "MyHelloWorld" in the **MIDlet-Name:** field.
3. Once you have entered all the needed information in the **Create Application Package** dialog, click **Generate...**.
4. Accept the location suggested (`Eclipse_workspace\MyHelloWorld\`) by Eclipse for the JAR and JAD files that will be generated by clicking **Save**.

Results

The `MyHelloWorld.jar` and `MyHelloWorld.jad` files needed to run the application on the emulator have now been created in the locations defined in the step above.

2.6.7 Running the MyHelloWorld MIDlet on the S60 emulator

Context

Viewing your MyHelloWorld MIDlet in the S60 emulator consists of running the JAD file that you have created .

Steps

1. Select **Tools > Carbide.j > Start Emulators...** from the Eclipse menu bar.
2. The **Start Emulators** pane is displayed:

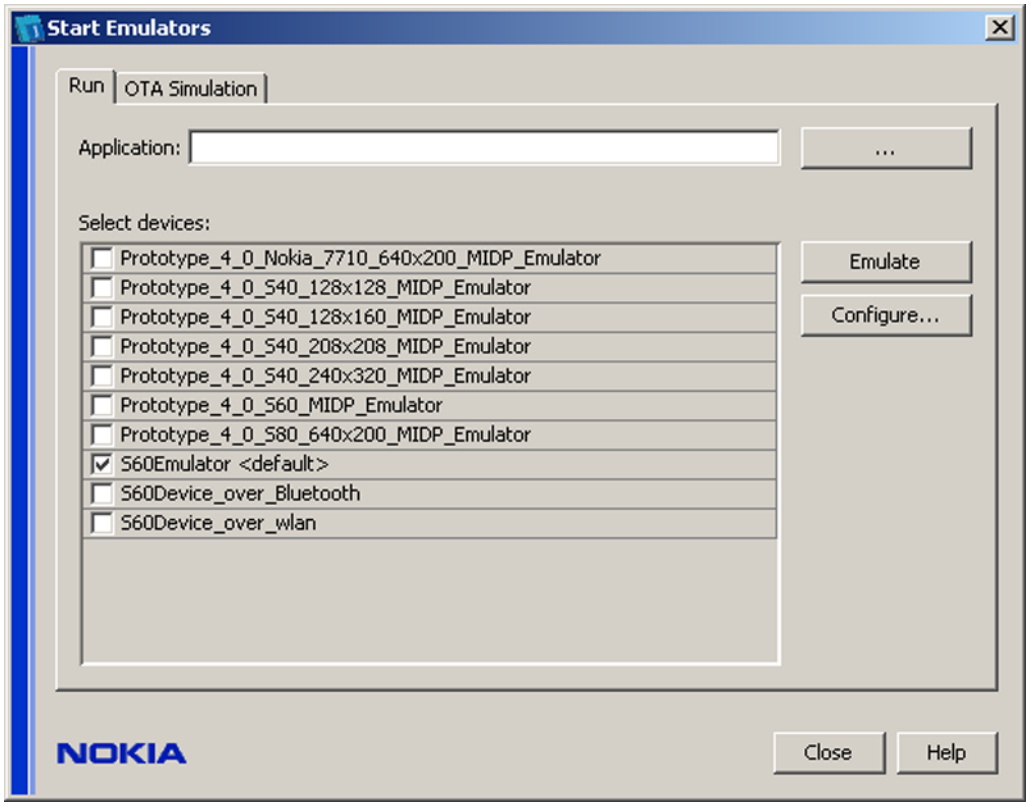


Figure 15: Start Emulators pane

2. In the **Start Emulators pane** pane, click the **...** button.
3. In the **Open** dialog, browse to the directory to which the JAD and JAR files of the MIDlet were created.

   For example: `C:\Eclipse_workspace\MyHelloWorld\`.
4. Select the `MyHelloWorld.jad` file and click **Open**.
5. The `MyHelloWorld.jad` file is opened into the **Start Emulators** dialog, as displayed in the **Application:** field

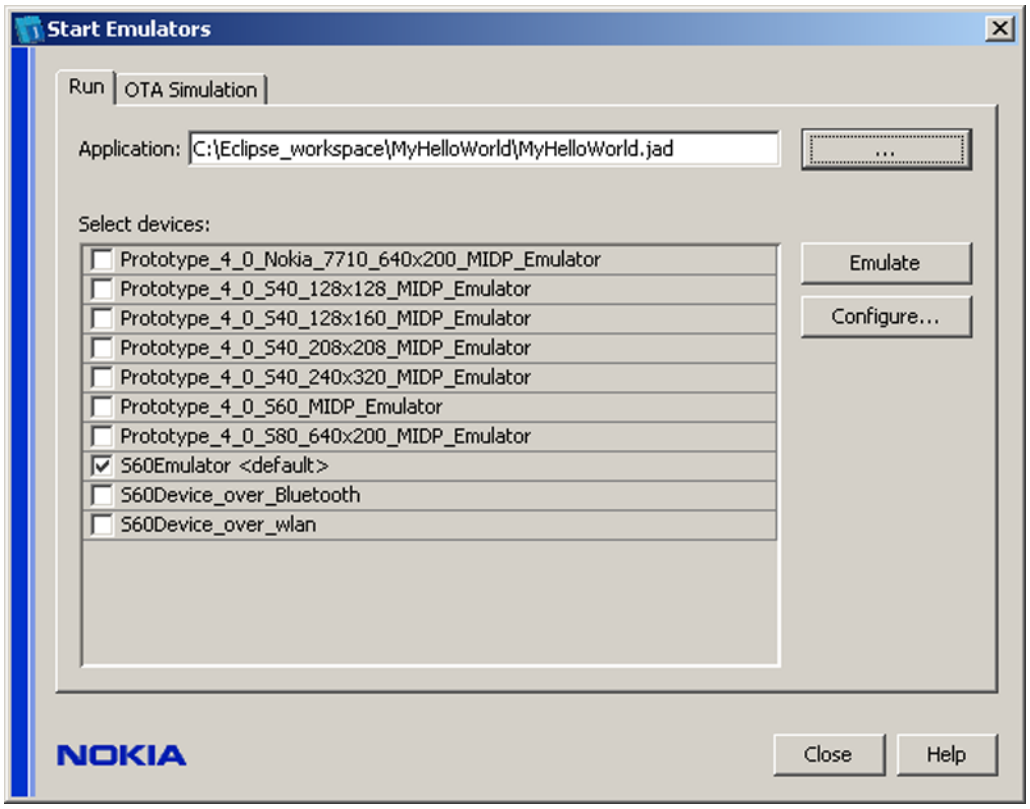


Figure 16: MyHelloWorld.jad file in Application field

5. Click the **Configure…** button if you want to define tracing output, debug and heap size options before running the application in the emulator.
6. Click the **S60Emulator <default>** checkbox in the **Select devices** list.
7. Click **Emulate** in the **Start Emulators** pane.

Results

The S60 emulator is launched.

Note: It may take some time for the emulator to start.

To locate the MyHelloWorld application, do the following:

1. Click the Applications button to open the emulator's application grid (see Graphical user interface).
2. In the emulator's application grid, use the five-way navigation key to open the **Installed** folder.
3. In the **Installed** folder, open the MyHelloWorld application.

2.7 Example: Creating an application from the command line interface

2.7.1 Prerequisites

2.7.1.1 What you will need

You should have the following tools and resources available on your PC before you can start developing Java MIDP applications using the S60 SDK:

- Windows 2000 (SP4 or higher) or Windows XP (SP2a or higher).

- Java 2 Standard Edition (J2SE) SDK version 1.4.2 or higher. The J2SE SDK provides the Java compiler and other tools used in application development. You can download the J2SE SDK from the Sun Developer Network website at http://java.sun.com/javase/downloads/index.jsp.
- S60 3rd Edition SDK for Symbian OS Supporting Feature Pack 1, for MIDP.
- A text editor, such as WordPad.

Notice that you should also check the application development environment before starting to develop applications from the command line interface.

2.7.1.2 Checking the application development environment

When you start developing an application from the Command Line Interface for the first time with the S60 SDK, you should first check that the development environment has been set up properly. This involves, for example, checking that the paths for the Java compiler and preverifier tool are set correctly.

If the paths have not been set correctly, you may receive the following error message:

"The name specified is not recognized as an internal or external command, operable program or batch file."

This means that Windows cannot locate the Java compiler (`javac.exe`) or the preverifier tool (`preverify.exe`) needed in compiling and preverifying the classes (see Building and running the Hello World Plus MIDlet) of the MIDlet.

Please refer to Troubleshooting: PATH variable not set or not set correctly for instructions on how to verify that the needed PATH variables have been set correctly.

Notice, that you can run the Java 2 SDK without setting the PATH variable. However, setting the PATH variable allows you to run the executables (`javac.exe`, `preverify.exe`) from any directory without having to specify the full path to the executable every time you run it. See Setting the PATH variable for detailed instructions on how to set the paths.

To check that the paths have been set correctly, please refer to Checking the paths for the Java compiler and preverifier tools.

2.7.1.3 Setting the PATH variable

Context

To avoid having to enter the full path every time you need to compile or preverify a MIDlet, it is useful to set the path permanently so it will persist even after rebooting. This is done by adding the full paths of the Java compiler and preverifier directories to the PATH variable.

Typically, this full path is `C:\j2sdk1.4.2_<version>\bin` for the Java compiler and `C:\<S60_SDK_installation_directory>\bin` for the preverifier.

Complete the following steps to set the path permanently on Microsoft 2000 and XP:

Steps

1. Choose **Start > Control Panel > System**.
2. Go to the **Advanced** tab and click the **Environment Variables** button.
3. In the **Environment variables** window, locate the **Path** variable in the **User Variables** or **System Variables** list.

   If you are not sure where to add the path, enter it as the last Path variable value listed in **User Variables**.

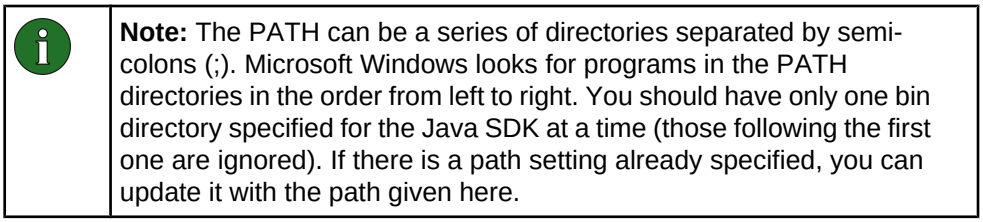
4. Select the **Path** variable and click **Edit**.

5. Enter the Java compiler and preverifier paths into the **variable value** field, for example, as follows:

   `C:\j2sdk1.4.2_<version>\bin` for the compiler.

   `C:\<S60_SDK_installation_directory>\bin` for the preverifier.

   Warning: Do not delete the existing path settings. Deleting the settings may affect the functioning of the system or other programs!



Note: The PATH can be a series of directories separated by semi-colons (;). Microsoft Windows looks for programs in the PATH directories in the order from left to right. You should have only one bin directory specified for the Java SDK at a time (those following the first one are ignored). If there is a path setting already specified, you can update it with the path given here.

6. Click **OK**.

7. In the **System Properties** window, click **OK**.

Results

The new paths have now been set for each time you open the command prompt.

You can verify the settings by completing the steps detailed in Checking the paths for the Java compiler and preverifier tools.

2.7.1.4 Checking the paths for the Java compiler and preverifier tools

Steps

1. In the Windows **Start** menu, open the Command Prompt by choosing **Start > Programs > Accessories > Command Prompt**.
2. To check the Java compiler (`javac.exe`) path settings, enter the following command in the Command Prompt:

   ```
   javac
   ```

   You should get the following output on the screen:

   ```
   Usage: javac <options> <source files> etc ... (output data)
   ```

   If you get an error message, the path has not been set correctly. See Troubleshooting: PATH variable not set or not set correctly for more details.
3. To check the preverifier tool (`preverify.exe`) path settings, enter the following command in the Command Prompt:

   ```
   preverify
   ```

   You should get the following output on the screen:

   ```
   Usage: preverify [options] classnames|dirnames ... etc ...
   (output data)
   ```

   If you get an error message, the path has not been set correctly. See Troubleshooting: PATH variable not set or not set correctly for more details.

Related information

- Setting the PATH variable

2.7.1.5 Troubleshooting: PATH variable not set or not set correctly

When testing the application environment settings, you may get the following error message (in Windows 2000, XP):

"The name specified is not recognized as an internal or external command, operable program or batch file"

This means that Windows cannot locate the Java compiler (`javac.exe`) or the preverifier tool (`preverify.exe`). See examples below on how to specify their locations for Windows.

To specify the location of the Java compiler:

If you installed the Java 2 Software Development Kit in, for example, `C:\jdk1.4`, enter the following command in the Command Prompt:

```
C:\jdk1.4\bin\javac
```

To specify the location of the preverifier tool:

For example, `C:\<S60_SDK_installation_directory>`, enter the following command in the Command Prompt:

```
C:\<S60_SDK_installation_directory>\bin\preverify
```

Note that in this case each time you compile or preverify an application, you have to precede your `javac` and preverify commands with `C:\jdk1.4\bin\` or `C:\<S60_SDK_installation_directory>\bin\`, such as, `C:\j2sdk1.4.2_<version>\bin\javac MyClass.java`. To avoid this, update the PATH variable as instructed in Setting the PATH variable.

2.7.2 Building and running the Hello World Plus MIDlet

This section describes how compile, preverify and run the simple HelloWorldPlus example MIDlet.

Compiling the HelloWorldPlus MIDlet

To compile the HelloWorldPlus example MIDlet, do the following:

1. In the Windows **Start** menu, open the Command Prompt by choosing **Start > Programs > Accessories > Command Prompt**.
2. In the Command Prompt, use the `mkdir` command to create a directory into which the Java compiler should output the classes created in compiling the MIDlet.

   Enter the following command in the Command Prompt:

   ```
   mkdir C:\tmpclasses
   ```
3. In the Command Prompt, use the `cd` command to switch to the `s60examples\` subdirectory of the directory where you installed the SDK:

   ```
   cd C:\<S60_SDK_installation_directory>\s60examples
   ```
4. Compile the Hello World MIDlet by entering the following command in the Command Prompt:

   ```
   javac -d C:\tmpclasses -classpath C:
   \<S60_SDK_installation_directory>\lib\hmidps60v31.jar C:
   \<S60_SDK_installation_directory>\s60examples\helloworldplus
   \src\com\nokia\midp\examples\lcdui\helloworldplus\*.java
   ```

Note: The PATH variable must be set correctly for the Java commands to work in the command prompt. See Setting the PATH variable for details.

The HelloWorldPlus MIDlet has now been compiled to the `tmpclasses` directory. Next, you will need to preverify the classes that have thus been created.

Preverifying the compiled HelloWorldPlus MIDlet

To allow more efficient memory usage on the target device, checking the code for correctness is done offline by using the preverifier tool. The preverifier checks that the classes are correct and inserts additional information into the class files. The Virtual Machine looks for this information and uses it for a much simpler verification based on the data collected by the preverifier.

To preverify the HelloWorldPlus MIDlet that you have compiled, do the following:

1 In the Windows Start menu, open the Command Prompt by choosing **Start > Programs > Accessories > Command Prompt**.

2 In the Command Prompt, use the `mkdir` command to create a directory into which the preverifier tool should output the classes created in compiling the MIDlet.

Enter the following command in the Command Prompt:

```
mkdir C:\classes
```

3 In the Command Prompt, use the `cd` command to switch to the `\bin` subdirectory of the directory where you installed the SDK:

```
cd C:\<SDK_installation_directory>\bin
```

4 Preverify the Hello World Plus MIDlet classes by entering the following command in the Command Prompt:

preverify -d c:\classes -classpath `C:\<S60_SDK_installation_directory>\lib\hmidps60v31.jar C:\tmpclasses`

Now that you have compiled and preverified the HelloWorldPlus MIDlet, you can run and view it in the emulator.

Running the HelloWorldPlus MIDlet in the emulator

Once you have compiled and preverified the HelloWorldPlus MIDlet, you can run and view it in the S60 emulator.

1 In the Windows **Start** menu, open the Command Prompt by choosing **Start > Programs > Accessories > Command Prompt**.

2 In the Command Prompt, use the `cd` command to switch to the `bin\` subdirectory of the directory where you installed the SDK.

Enter, for example, the following command in the Command Prompt:

cd `C:\<S60_SDK_installation_directory>\bin`

3 Run the Hello World Plus MIDlet in the emulator by entering the following command in the Command Prompt:

```
emulator -classpath C:\classes
com.nokia.midp.examples.lcdui.helloworldplus.HelloWorldPlusM
IDlet
```

The emulator is opened:

Figure 17: S60 emulator

Note: It may take some time for the emulator to start.

To locate the MyHelloWorld application, do the following:

1 Click the Applications button to open the emulator's application grid (see Graphical user interface).

2 In the emulator's application grid, use the five-way navigation key to open the **Installed** folder.

3 In the **Installed** folder, open the MyHelloWorld application.

To simulate pressing buttons on a real device, click the corresponding keys on the emulator with the mouse (for more information on emulator functionality, please refer to the Emulator Guide). Notice, that the **Close** and **Exit** commands also close the emulator.

Now that you have you have compiled and preverified the Hello World Plus MIDlet and viewed it in the S60 emulator, you can install it to a target device. Before doing this you can, however, personalize the application. These procedures are covered in Creating your own application from the Hello World Plus MIDlet.

2.7.3 Creating your own application from the Hello World Plus MIDlet

Context

In this section you will tailor the Hello World Plus MIDlet into a new, personalized application of your own by editing the source code of the MIDlet.

In order to be able to edit the source code, you need to open it in an editor. To open the source in a WordPad editor, do the following:

Steps

1. In the Windows Start menu, open the Command Prompt by choosing **Start > Programs > Accessories > Command Prompt**.
2. In the Command Prompt, use the `cd` command switch to the `\s60examples\src\com\nokia\midp\examples\lcdui\helloworldplus` subdirectory of the directory where you installed the SDK.

   Enter, for example, the following command: `cd C:\<S60_SDK_installation_directory>\s60examples\src\com\nokia\midp\examples\lcdui\helloworldplus`
3. Open the HelloWorldPlus MIDlet source file in the WordPad editor by entering the following command: `write HelloWorldPlus.java`

   The `HelloWorldPlus.java` source file opens in WordPad:

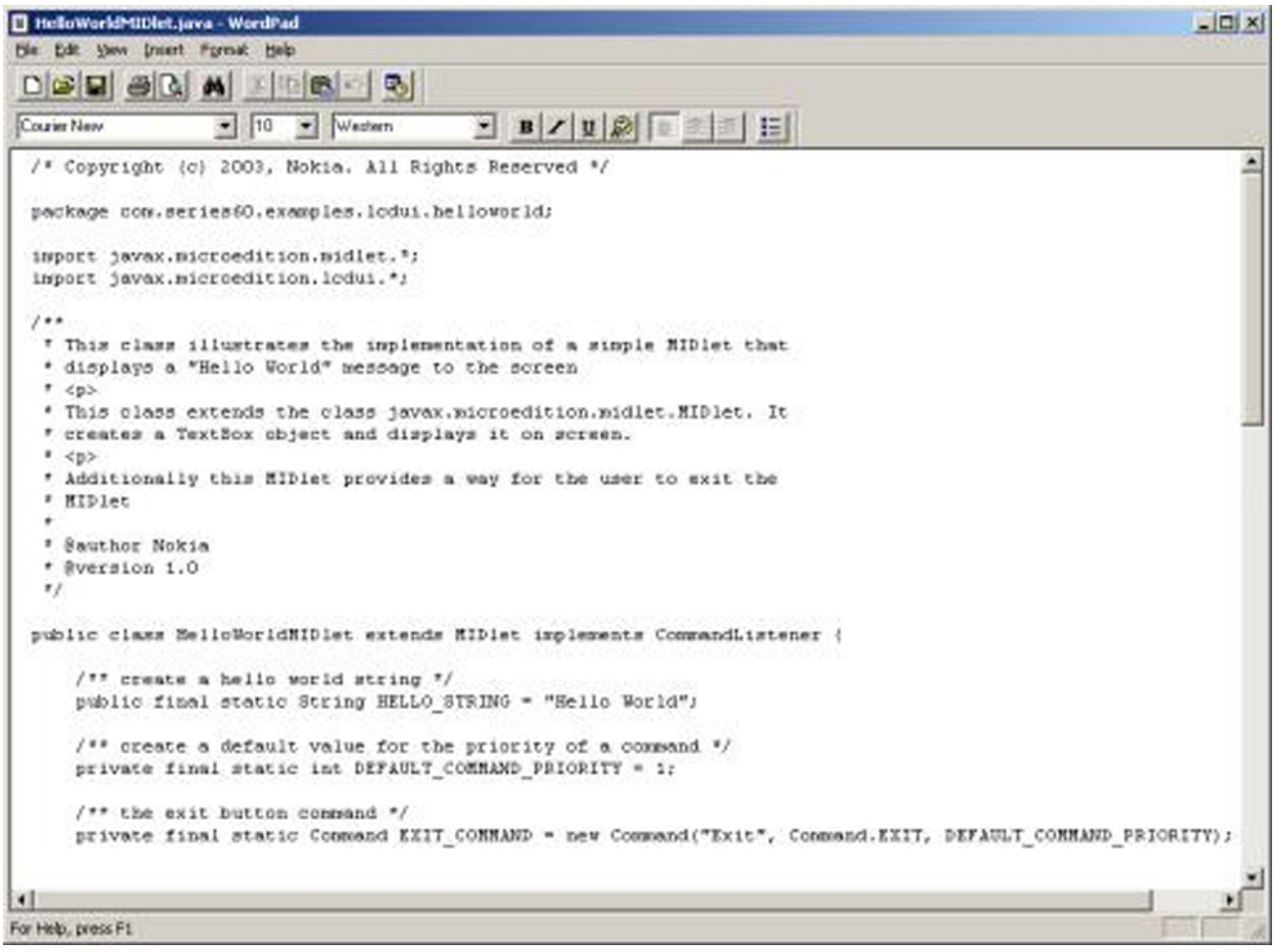


Figure 18: HelloWorldPlus MIDlet in WordPad editor

To change the text displayed in on the screen in the HelloWorldPlus application, you first need to locate it in the source file.

4. Select **Edit > Find** from the menu bar (or press **CTRL+F**).

   The **Find** dialog box is displayed.

5. In the **Find** dialog box, enter the text `HELLO_STRING = "Hello World!"` in the **Find what:** field and click the **Find Next** button.

   The cursor will go to the "Hello World" string in the `HelloWorldPlus.java` source file:

   ```
   public final static String HELLO_STRING = "Hello World!";
   ```

6. Close the **Find** dialog box by clicking **Cancel**.

7. Edit the source code, for example, as follows:

   ```
   public final static String HELLO_STRING = "Just Java it!";
   ```

8. To save your changes, select **File > Save** and click **Yes** to confirm the command.

9. Compile, preverify and run your new MIDlet as described in Building and running the Hello World Plus MIDlet.

	**Note:** If you have already created the `tmpclasses` and `classes` directories, you can skip the `mkdir` commands when compiling, preverifying and running your MIDlet.

Results

Once you have compiled and preverified the new MIDlet, you can run it in the S60 emulator. The emulator should now display the text "Just Java It!"

Once you have tested the application on the emulator and you are satisfied with it, the next step is to package your MIDlet as described in Installing your application to your S60 smartphone.

2.7.4 Installing your application to an S60 smartphone

Context

Installing a MIDlet to an S60 smartphone involves transferring the `.jad` and `.jar` files which you have created (see Creating your own application from the Hello World Plus MIDlet) to your S60 smartphone. Follow the instructions provided in your smartphone's user manual on how to install new applications as the installation procedure varies depending on the device. The steps below provide an example on how to install a MIDlet by using a Bluetooth connection.

	**Note:** Installing the MIDlet requires that your smartphone supports the open Java™ ME MIDP platform.

In the following, an example of how to install a MIDlet to your S60 smartphone is provided.

For other installation options, please refer to Deploying MIDlets to a device .

Steps

1. Make sure that Bluetooth connectivity is activated on your computer.
2. Enable Bluetooth connectivity on your S60 smartphone.
3. Open the Windows Explorer by selecting **Start > Programs > Accessories > Windows Explorer** on your computer.

4. Go to the `classes\` subdirectory of the directory where you created the JAR and JAD files `HelloWorldPlus.jar` and `HelloWorldPlus.jad`.
5. Select the files `HelloWorldPlus.jar` and `HelloWorldPlus.jad`.
6. Right-click and select **Send To > Bluetooth > Other…**.
7. Select **Find Devices**.
8. Select your S60 smartphone from the list of devices and click **OK**.

   The installation files are transferred to your S60 smartphone from a computer via Bluetooth. Your smartphone should receive the installation files as a message attachment.
9. Go to the messaging folder in your S60 smartphone and open the `HelloWorldPlus.jad` message.
10. If requested, confirm the installation command.

    A notification is displayed when the installation is complete.

Results

The HelloWorldPlus MIDlet is installed to your S60 smartphone. You can open the HelloWorldPlus Java application from your phone's application grid.

3 Tools and Utilities

3.1 Emulator Guide

3.1.1 S60 emulator

The S60 SDK emulator enables you to view and test applications on your PC before installing them to a real device. It provides a graphical interface of a real phone with the needed phone functionality to test your application. The emulator mimics the operation of an application on a real phone so accurately that application development can be done even before the required hardware, that is, an S60 smartphone, is available.

This section provides information and instructions on how to use the emulator in S60 application development.

Figure 19: S60 SDK emulator

3.1.2 Using the Emulator

3.1.2.1 Starting the emulator

3.1.2.1.1 Starting the S60 emulator from the Windows Start menu

Steps

1. From the **Start** menu, select **Start > All Programs > S60 Developer Tools > 3rd Edition FP1 SDK > MIDP > Emulator**.

Results

The emulator starts:

Figure 20: S60 SDK emulator

3.1.2.1.2 Starting the S60 emulator from the command line

Steps

1. Use the `CD` command and then enter the SDK installation directory into the Command Prompt:

   ```
   CD C:\<SDK_installation_directory>\bin\epoc32\release\winscw
   \udeb
   ```

2. To this command add: **`sdk.exe`**. The complete command should look like this:

   ```
   C:\<SDK_installation_directory>\bin\epoc32\release\winscw
   \udeb>sdk.exe
   ```

Results

The emulator starts:

Figure 21: S60 SDK emulator

	Note: For more detailed information about using the emulator from command line, including syntax and arguments, please refer to Command line interface.

3.1.2.1.3 Starting the S60 emulator from an IDE

The S60 SDK supports the following IDEs:

- Carbide.j 1.5
- NetBeans 5.0
- Eclipse 3.1.2
- IBM WebSphere Studio Device Developer 5.7.1

Starting the emulator from an IDE typically happens by fist configuring the emulator for the IDE in question and then running an application from the IDE. This will open the emulator, in which you can then view the application in question.

Notice that the emulator is only able to run applications packaged into a MIDlet Suite as specified in the MIDP specification. This means that both a JAD file (Java Application Descriptor) and a JAR file (Java Archive) must be created before running the application.

For more information on how to run applications from supported IDEs, please refer to the following topics:

- Running an S60 SDK example application from Carbide.j

- Running an S60 SDK example application from NetBeans
- Launching a MIDlet in the S60 emulator from WSDD

3.1.2.2 Using the keyboard

This section describes how to use the keyboard for inputting text and provides a list of the keyboard shortcuts and function keys available in the S60 emulator.

In addition to using the keys of the emulator's graphical user interface, you can also use the keyboard of your PC for entering data in text boxes when the emulator is in the edit mode (that is, a pen followed by the characters "abc" appears in the status pane). You can use all the alphanumeric keyboard keys (**a**-**z**, **0**-**9**) and the following non-aphanumeric characters: **! " % & / ( ) = ? ` < > | , . ; : _ - ~ ^ ¨ ¨ [ ]**. A long press of the key will display the corresponding number.

The following table lists the emulator function keys and keyboard shortcuts for:

- emulating mouse clicks on the emulator GUI
- resource allocation functions
- drawing functions.

Table 1.1: S60 emulator keyboard shortcuts

Key	Function
Esc	Cancels currently displayed dialogs or menus.
F9	Turns off the emulator. Re-press to restore the emulator.
F10	Simulates an emergency shutdown. Press F9 to restore.
The key below Esc (suc as §)	Corresponds to the Selection key, that is, OK on the navigation key.
Alt+1	Corresponds the left soft key.
Alt+2	Corresponds the right soft key.
Cursor keys	Correspond the navigation keys.
0-9	Correspond the ITU-T numeric keypad.
Home	Corresponds the Applications key.
Backspace	Correspond the Clear key.
Ctrl+Al+tShift+O	Rotates the Emulator skin by 180 degrees.
Ctrl+Alt+Shift+J	Displays coloured borders around controls.
Ctrl+Alt+Shift+P	Displays information pertaining to the resource failure tool.
Ctrl+Alt+Shift+Q	Turns off the heap failure mode.
Ctrl+Alt+Shift+R	Redraws the screen.
Ctrl+Alt+Shift+F	Enables the Window Server auto-flush.
Ctrl+Alt+Shift+G	Disables the Window Server auto-flush. (Default)
Ctrl+Alt+Shift+M	Displays a Move Me dialog.
Ctrl+Alt+Shift+Enter	Displays a Move Me dialog.

3.1.2.3 Changing the emulator resolution

The emulator supports the following configurations (resolutions):

- QVGA Portrait (240x320)
- QVGA Landscape (320x240)

You can change the resolution of the emulator while its running by:

- clicking the **Switch Configuration** button (see figure below) or

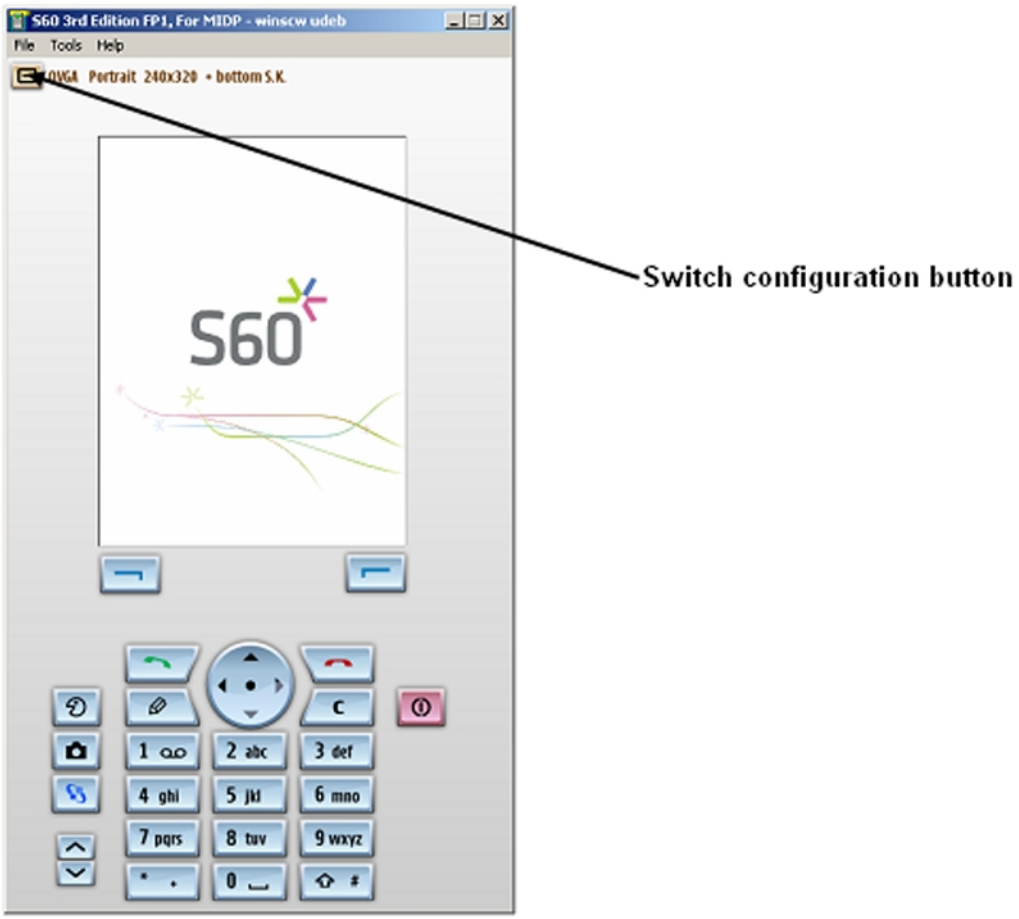


Figure 22: Changing the resolution while the emulator is running

- selecting **Tools > Switch Configuration** from the menu bar.

Related information

- Setting the default resolution of the emulator
- Scalable UI

3.1.2.4 Loading content to the emulator

3.1.2.4.1 Opening files in the emulator

You can open various different kinds of files in the emulator through the emulator's **File > Open...** option. The supported file formats are:

- HTML
- XHTML
- WML
- MMS files
- MIDP files
- Image fies:
  - `.wmf`
  - `.bmp`
  - `.gif` (animated `.gif` also supported)
  - `.tif`

  - `.png`
  - `.mbm`
  - `.jpg`
- WAP push files:
  - `.sic` (Service Indication Compiled)
  - `.slc` (Service Loading Compiled)
- DRM files:
  - Forward lock (`.dm`)
  - Combined Delivery (`.dm`)
  - Separate Delivery content (`.dcf`)
  - Encoded Rights for Separate Delivery content (`.drc`)

3.1.2.4.2 Opening a web page in the emulator

To open and view a web page in the emulator, do the following:

1 In the emulator menu bar, select **File > Open URL...**.

2 In the Open URL dialog that opens, type the URL address of the web page that you want to view into the **Open:** field (or select one from the drop-down menu) and click **OK**.

3 The URL selected is displayed in the emulator.

	**Note:** You can also open files through this dialog by clicking the **Browse** button and selecting a file from the **Open** dialog that is displayed.

3.1.2.4.3 Loading content from the NMIT

You can load the following content to the emulator from the Nokia Mobile Internet Toolkit (NMIT) content creation tool.

- XHTML
- WML
- MMS
- WAP Push
- DRM
- Download descriptor.

	**Note:** Unlike in actual devices, the SDK requires that DRM rights be loaded before loading the corresponding dcf content to the SDK.

WAP Push Service Indication (`.sic`) and DRM Rights (`.drc`) cannot be pushed from NMIT to the SDK. The workaround is:

1 Save the SIC and DRC content in NMIT by using the **Save Binary** option.

2 In the SDK, select **File > Open** to load the Rights objects.

Forward Lock/Combined Delivery (`.dm`) or Separate Delivery (`.dcf`) content can both be loaded from NMIT to the SDK or selected through **File > Open** in the SDK.

3.1.2.5 Event simulation

3.1.2.5.1 Events tab

The **Events** tab of the Utilities (**Tools > Utilities**) window provides an interface for simulating MMC hotswap and various different kinds of phone events in the emulator.

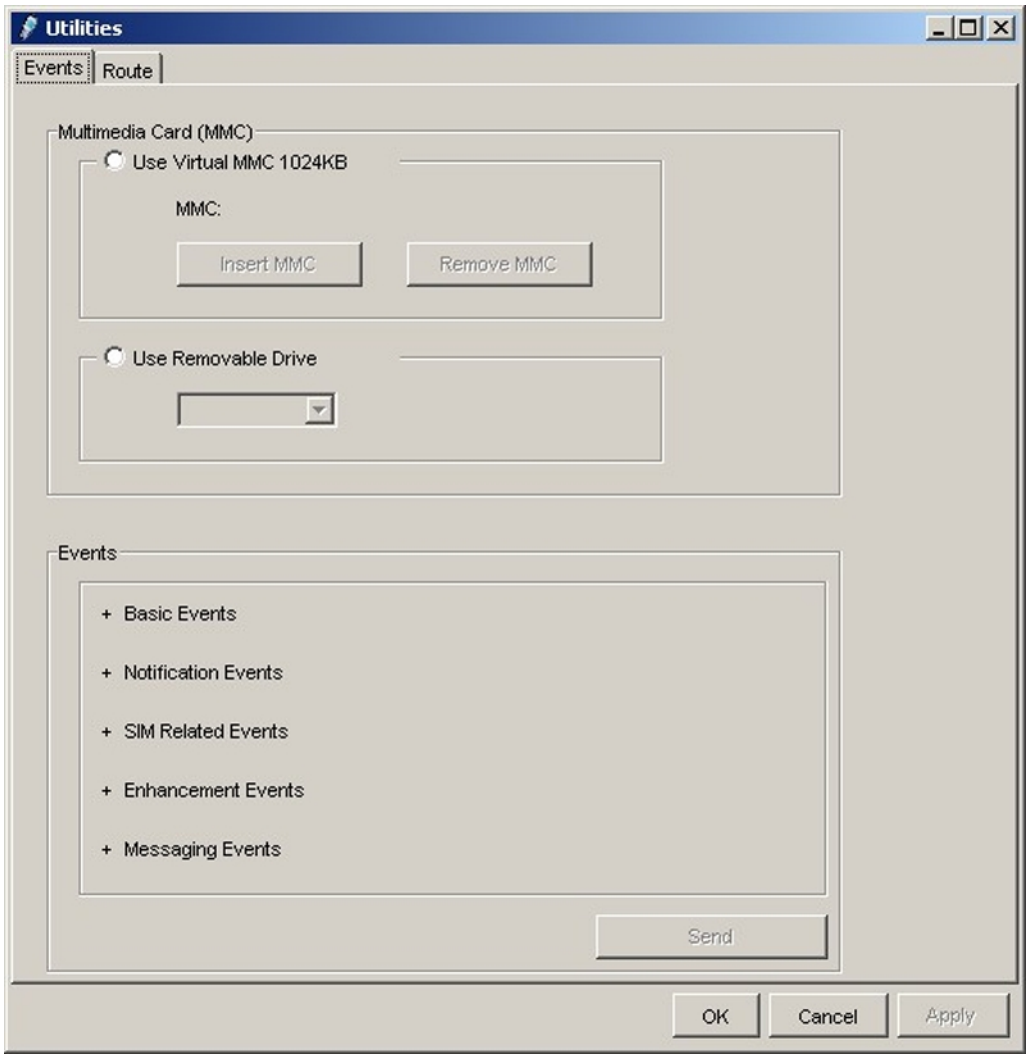


Figure 23: Events tab in the Utilities window

The **Utilities** window contains the follwoing two main sections:

- **Multimedia Card (MMC)** - Allows you to select whether to use a virtual MMC or map one of the removable drives of the workstation to be used as an MMC drive in the emulator.
- **Events** - Allows you to simulate various phone events in the emulator. The Event parameter input fields are event-specific and shown when an event is opened by clicking the + button. You can define the delay in seconds for the selected event in the Delay field. The default value for the delay is 1 second. To submit a selected event, click the check box of the event that you wish to emulate and then click the Send button.

The **OK**, **Cancel** and **Apply** buttons function as follows:

- **OK** - Saves the modified selection and closes the window.
- **Cancel** - Exits the window without saving the settings
- **Apply** - Saves the modified settings without exiting the window.

Related information

- Simulating MMC hotswap
- Simulating basic events
- Simulating notification events
- Simulating SIM-related events

- Simulating enhancement-related events
- Simulating messaging events

3.1.2.5.2 Simulating MMC hotswap

You can select whether to use a virtual MMC or map one of the removable drives of the workstation to be used as an MMC drive in the emulator.

	**Note:** Changing between the MMC emulation types requires confirming the selection by pressing the **OK** button and restarting the emulator.

Use Virtual MMC

The **Use Virtual MMC** option allows you to use a virtual MMC. Use the **Insert MMC** and **Remove MMC** buttons to simulate MMC card removal and insertion.

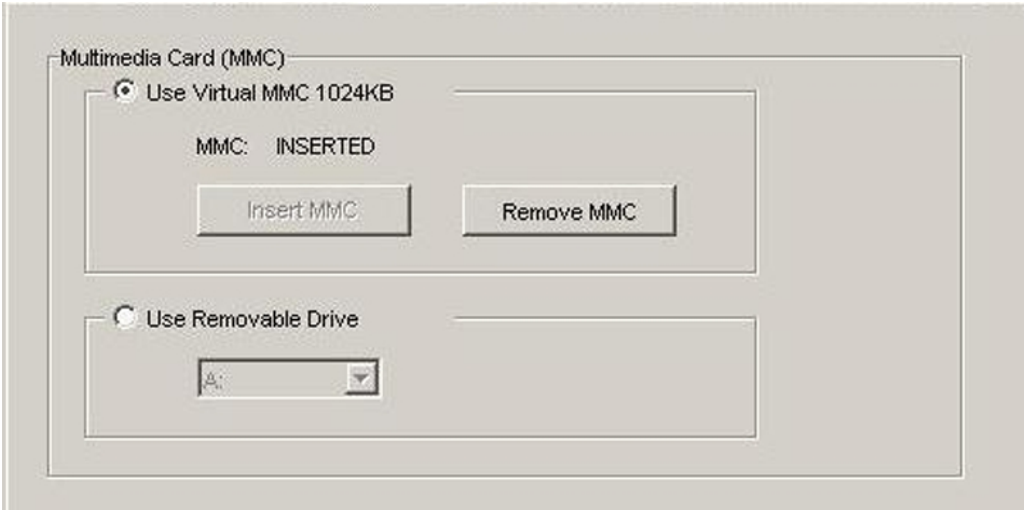


Figure 24: Use Virtual MMC

Use Removable Drive

With the **Use Removable Drive** option you can map a removable drive to be used as the MMC drive. Select the drive from the drop-down menu.

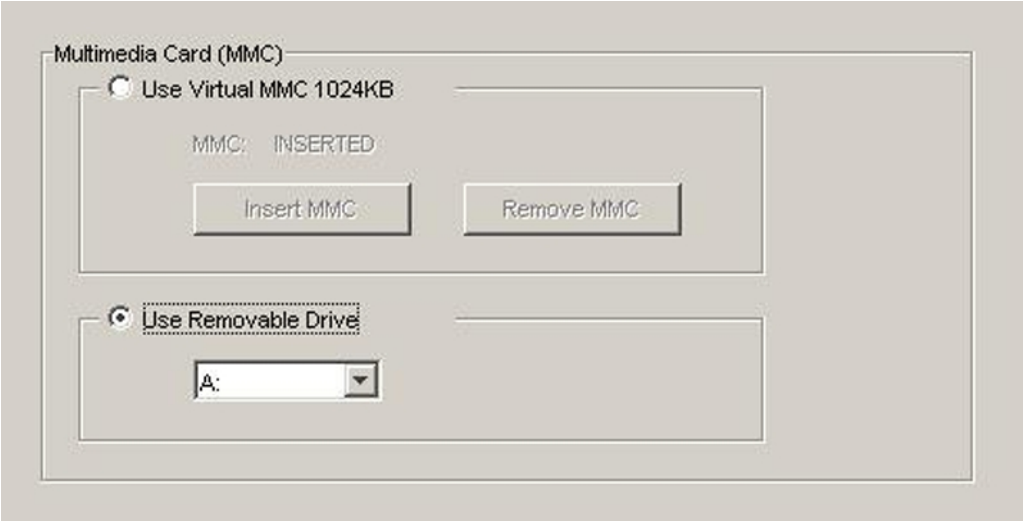


Figure 25: Use Removable Drive

3.1.2.5.3 Simulating basic events

The **Basic Events** event type includes the following events:

- **Grip Open** - simulates a case when the grip of the phone is open.
- **Grip Closed** - simulates a case when the grip of the phone is closed.
- **Battery Low** - simulates a case when a phone battery has low power.
- **Battery Empty** - simulates a case when the phone battery runs out of power.
- **Connecting External PowerCharger** - simulates a case when the phone is connected to an external power charger.
- **Battery Full** - simulates a case when charging has ended and the phone displays the "Battery Full" message.
- **Removing External PowerCharger** - simulates a case when a charger is disconnected from the phone.

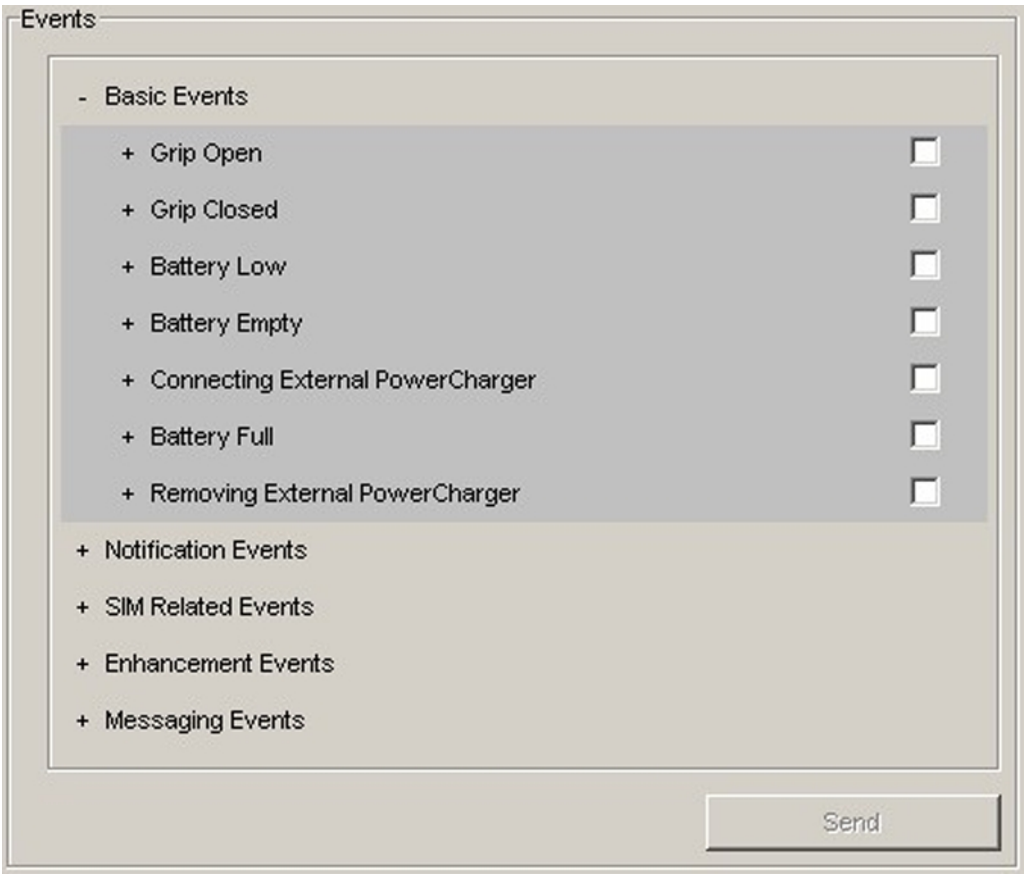


Figure 26: Basic Events

GUI Principles of the Basic Events

To open an event type, click the **+** button next to it. For example, the **Basic Events** event type contains the Grip Open event following events, which can be simulated in the emulator by selecting it from the list of events:

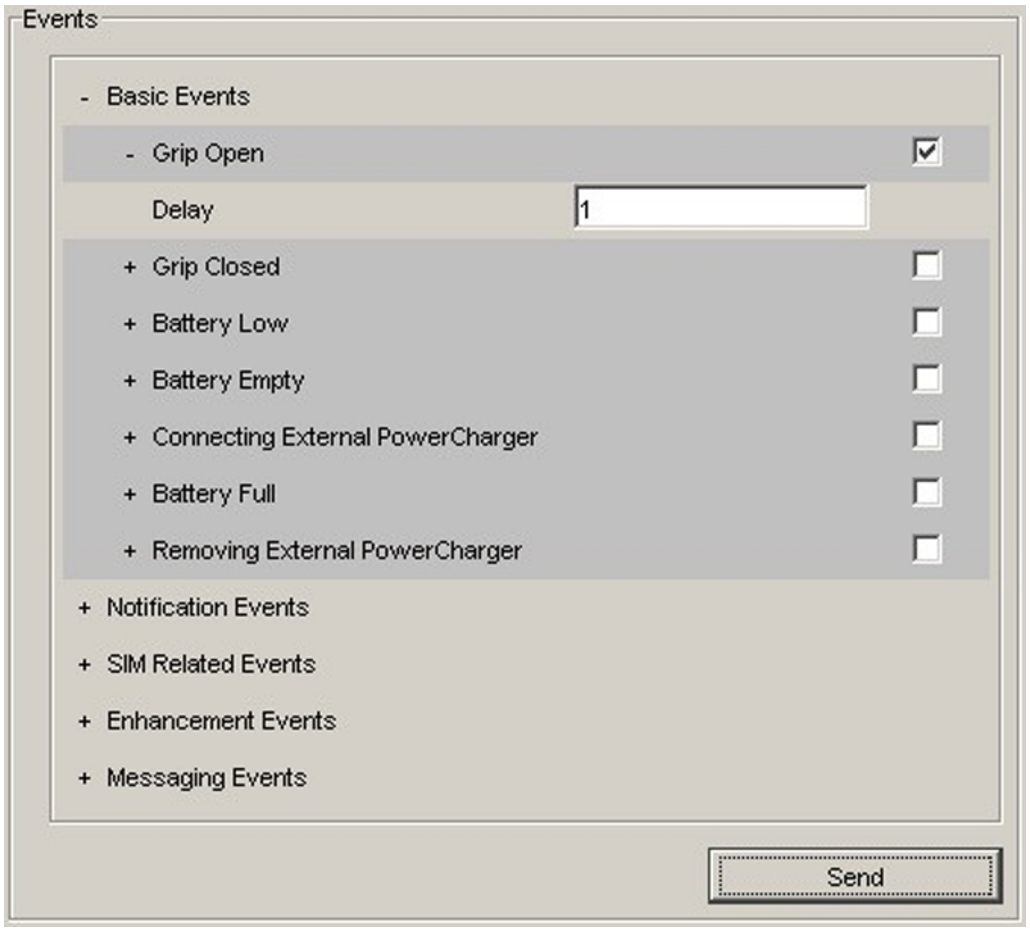


Figure 27: The parameter input fields of the Grip Open event

Event parameter input fields are event specific and shown when an event is opened by clicking the **+** button. You can define the delay in seconds for the selected event in the **Delay** field. The default value for the delay is 1 second.

To submit a selected event, click the check box of the event that you wish to emulate and click the **Send** button.

3.1.2.5.4 Simulating notification events

The **Notification Events** event type includes the Alarm notification event:

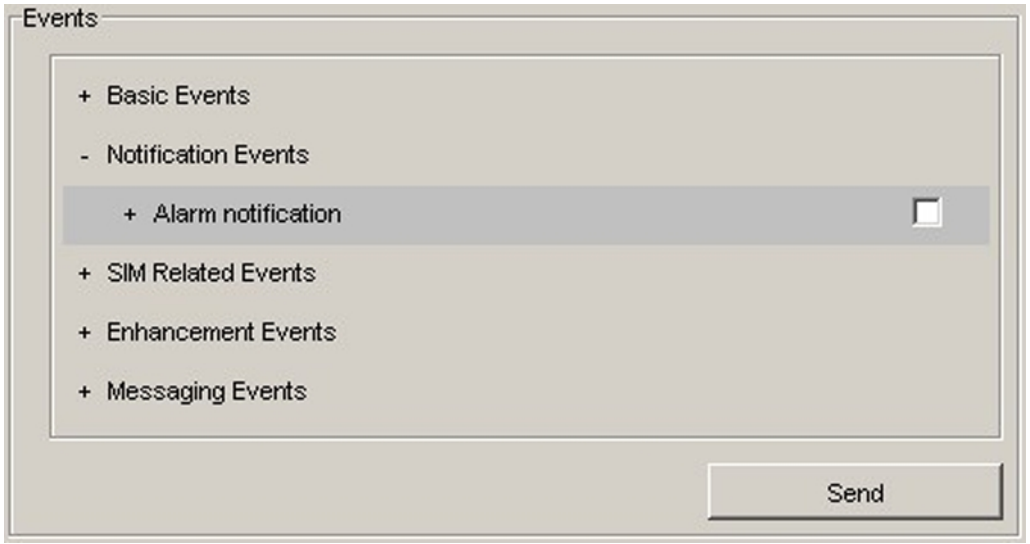


Figure 28: Notification Events dialog

The **Alarm notification** event simulates a case when the phone displays an alarm notification message.

- With the **Minutes from now** parameter you can define the moment of the alarm notification. The default **Minutes from now** value is 0.

To submit a selected event, click the check box of the event that you wish to emulate and click the **Send** button.

3.1.2.5.5 Simulating SIM-related events

The **SIM-related Events** event type includes the following events:

- **SIM Removed** - simulates a case when the SIM is removed.
- **SIM Locked** - simulates a case when the SIM is locked.
- **SIM SMS Memory Full** - simulates a case when the SIM SMS memory is full.

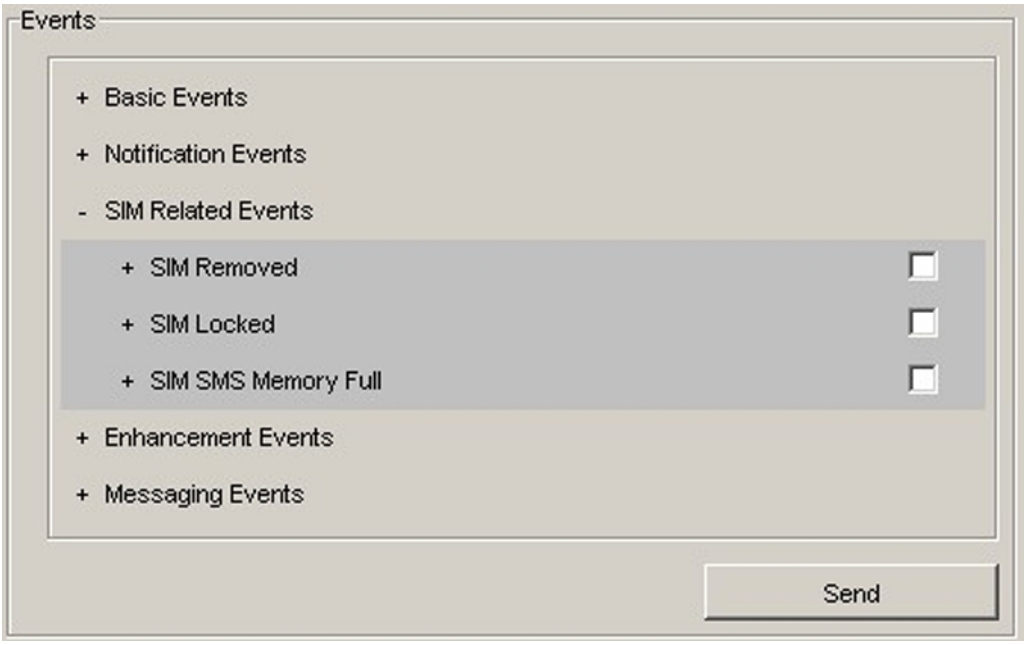


Figure 29: SIM-related events

You can define the delay in seconds for the selected event in the **Delay** field. The default value for the delay is 1 second.

To submit a selected event, click the check box of the event that you wish to emulate and click the **Send** button.

3.1.2.5.6 Simulating enhancement-related events

The **Events Related to Enhancements** event type includes the following events:

- **Headset connected** - simulates a case where a headset is connected to the phone and the headset indicator appears on the screen.
- **Headset removed** - simulates a case where a headset is removed from the phone and the headset indicator disappears from the screen.
- **TTY connected** - simulates a case where a tty is connected to the phone and the tty indicator appears on the screen.
- **TTY removed** - simulates a case where a tty is removed from the phone and the tty indicator disappears from the screen.
- **Loopset connected** - simulates a case where a loopset is connected to the phone and the loopset indicator appears on the screen.

- **Loopset removed** - simulates a case where a loopset is removed from the phone and the loopset indicator disappears from the screen.
- **Handsfree mode on** - simulates a case where the handsfree mode of a phone is on.
- **Handsfree mode off** - simulates a case where the handsfree mode of a phone is off.

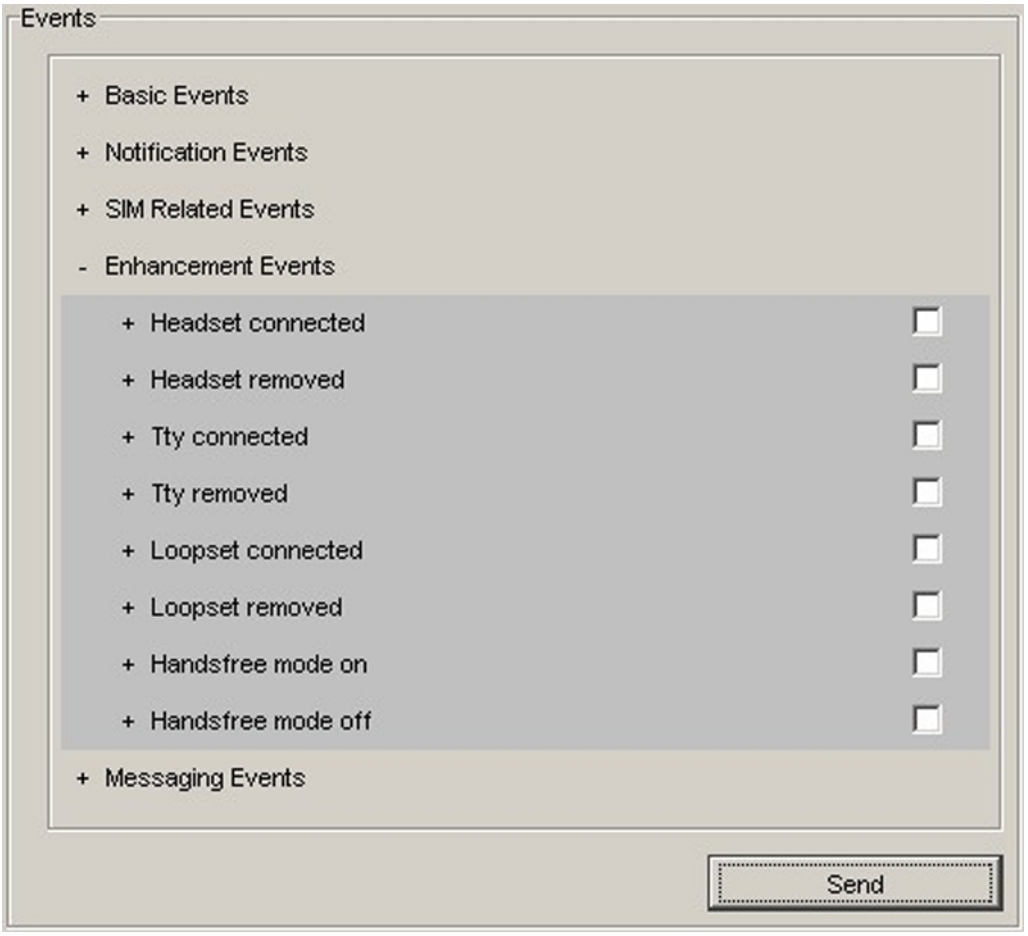


Figure 30: Enhancements Events dialog

You can define the delay in seconds for the selected event in the **Delay** field. The default value for the delay is 1 second.

To submit a selected event, click the check box of the event that you wish to emulate and click the **Send** button.

3.1.2.5.7 Simulating messaging events

The **Messaging Events** event type includes the following events:

- **SMS Message** - copies an example SMS message to the Inbox folder of the emulator.
- **MMS Message** - copies an example MMS message to the Inbox folder of the emulator.

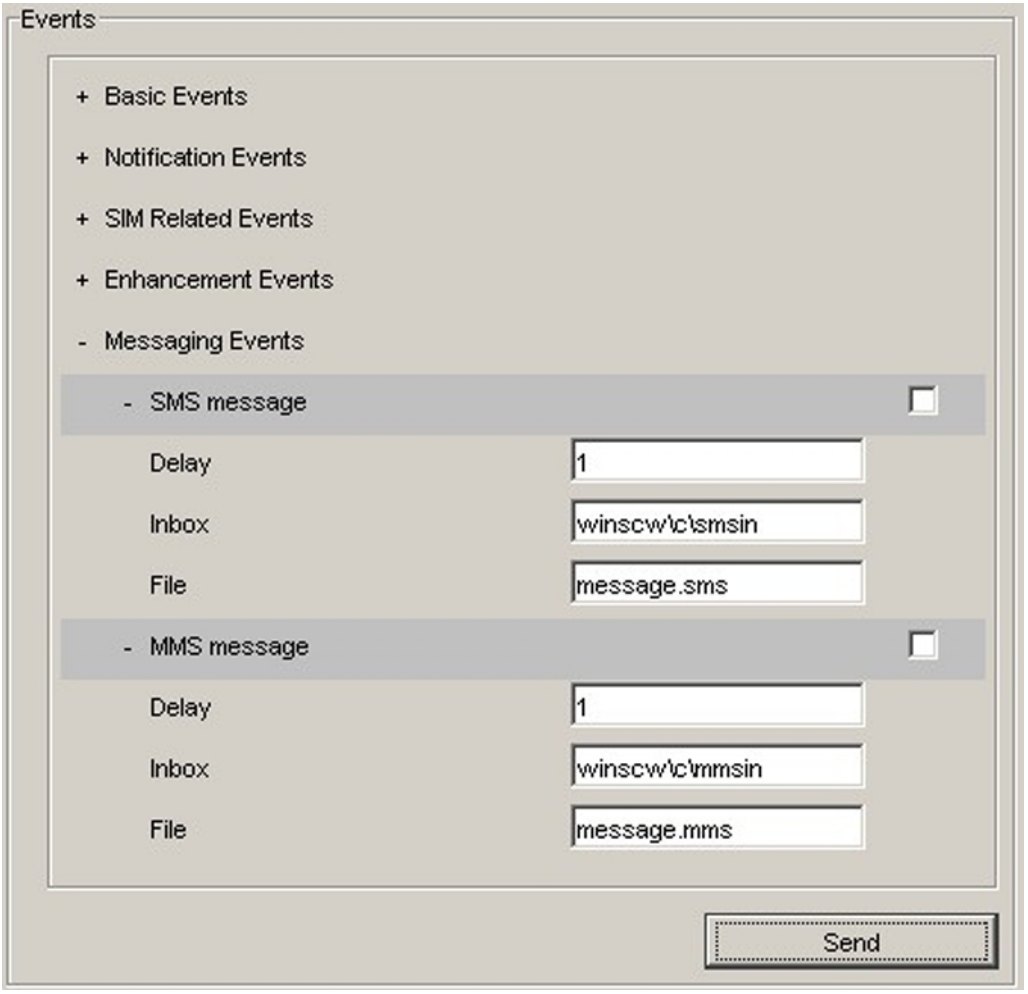


Figure 31: Messaging Events dialog

You can define the delay in seconds for the selected messaging event in the **Delay** field. The default value for the delay is 1 second.

	Note: Do not modify the contents of the **Inbox** and the **File** fields.

To submit a selected event, click the check box of the event that you wish to emulate and click the **Send** button.

3.1.2.6 Diagnostics and tracing

3.1.2.6.1 Diagnostics window panes

To utilize the diagnostics and tracing features of the SDK, open the Diagnostics window from the emulator by selecting **Tools > Diagnostics** from the emulator's menu bar.

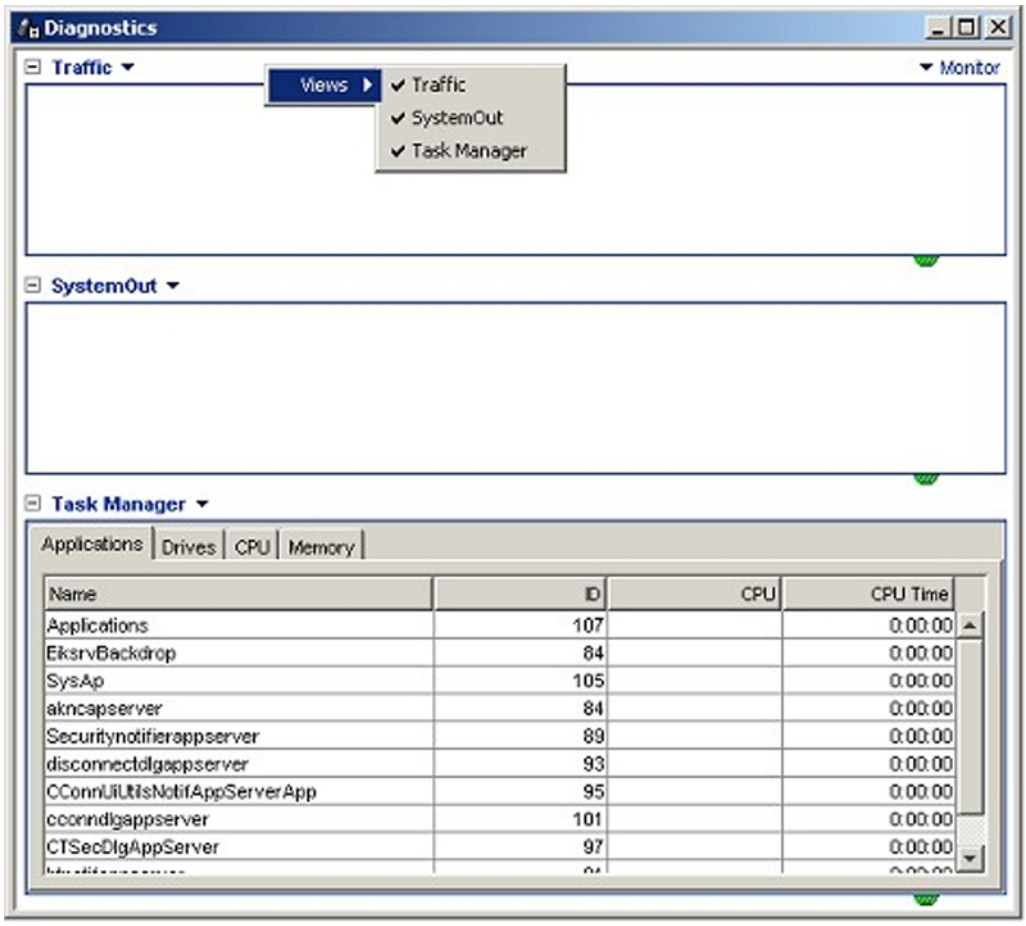


Figure 32: Diagnostics window

The **Diagnostics** window provides panes that display different diagnostics type data from the emulator or a device:

- Traffic pane
- SystemOut pane
- Task Manager pane

3.1.2.6.2 Traffic pane

The **Traffic** pane provides information on emulator http traffic (requests and responses). The pane has two menus: **Traffic** and **Monitor**. The tab also provides a context menu, which is opened by clicking the right mouse button in the pane.

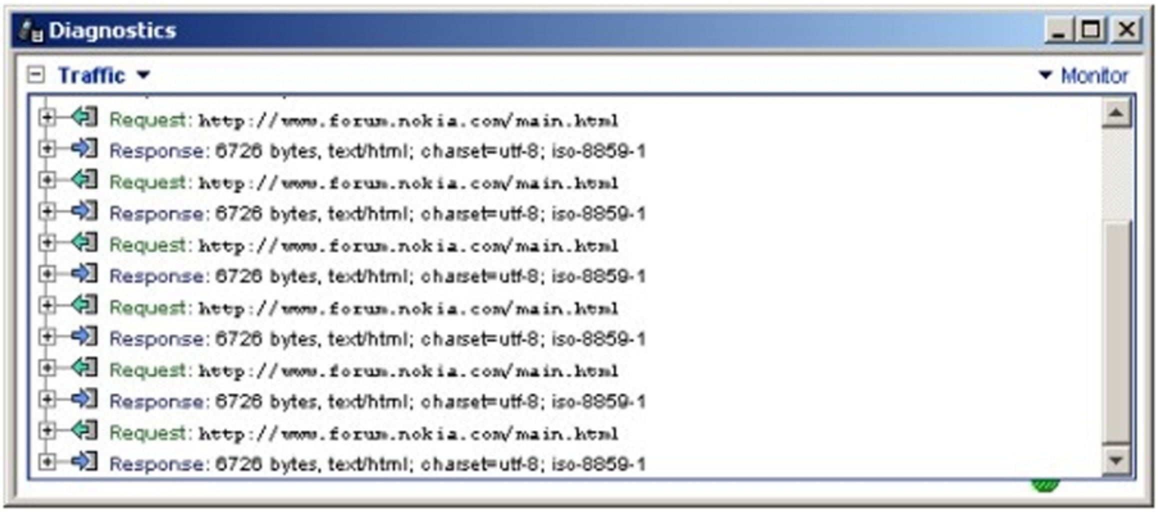


Figure 33: Traffic pane

The **Traffic** and **Monitor** menus can be opened by clicking their arrow down buttons and provide the following options:

Traffic menu items	Description
Clear traffic info	Removes all items from panel.

Monitor menu items	Description
Requests (Filtering option)	Selects whether to show or hide all http requests.
Content responses (Filtering option)	Selects whether to show or hide all content responses.
Select all / none	Selecting all or no items from panel.

The context menu of the Traffic pane can be opened by clicking the right mouse button in the pane. It provides the following options:

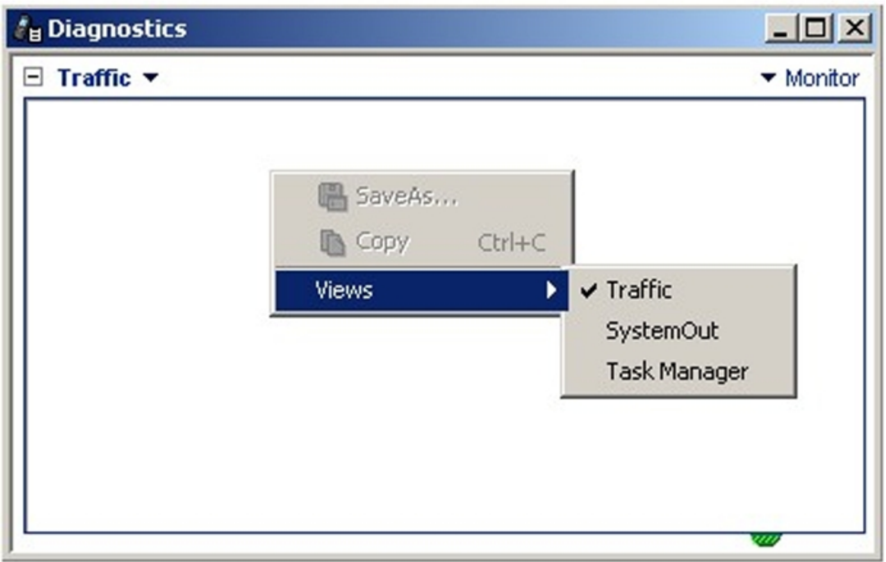


Figure 34: Traffic pane context menu

Context menu items	Description

Save As	Opens Save to file dialog where user can chose file name to save text output.
Copy	Copies grayed area to clipboard.
Views >	Provides the same options as in the general context menu, that is: - Traffic - SystemOut - Task Manager

3.1.2.6.3 SystemOut pane

The **SystemOut** pane is used as an output pane for messages from running applications. These messages can be sent from REcmt API or MIDlet System.out on page 0 redirection, when enabled.

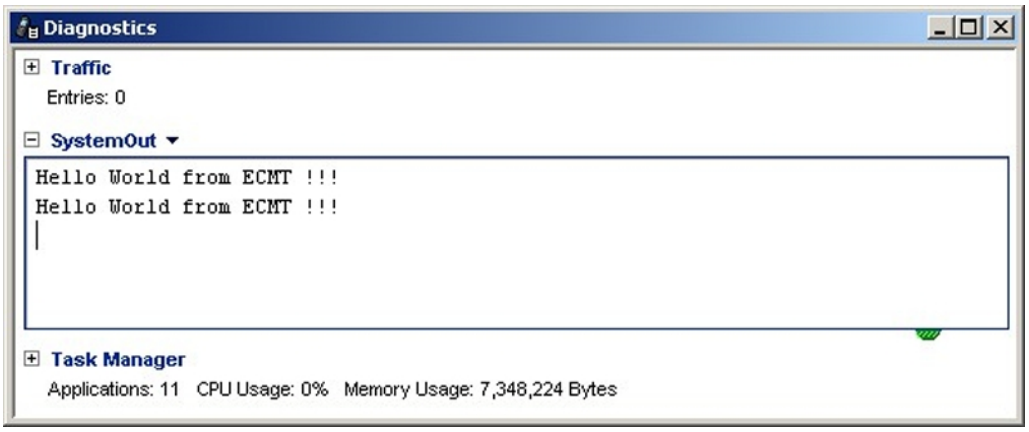


Figure 35: SystemOut pane

The **SystemOut** pane provides the following menu options, which can be accessed by clicking the arrow down button next to the pane caption:

Menu items	**Description**
Copy	Copies selected items from the panel to the clipboard.
Find	Finds utility for string sequences.
Clear	Clears items from the panel.
Log to file	Opens the file dialog, where you can set the output file name and location.

The menu options can also be accessed in the context menu, which is opened by clicking the right mouse button in the pane:

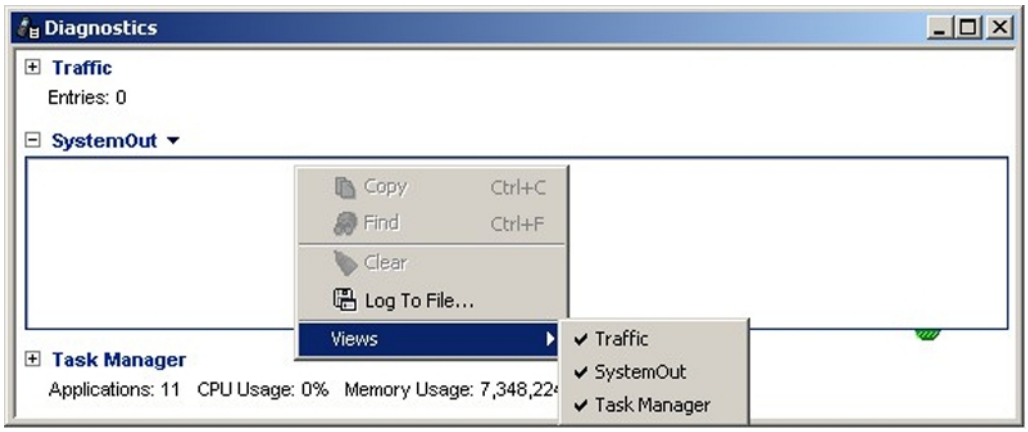


Figure 36: SystemOut pane context menu options

3.1.2.6.4 Task Manager pane

The **Task Manager** pane displays information of the emulator's runtime state. The minimized view of the tab displays current data in compact mode:

Figure 37: Task Manager pane minimized view

In addition to the minimized view, the **Task Manager** pane also contains a full mode, which can be opened by clicking the + button on the left side of the **Task Manager** pane title.

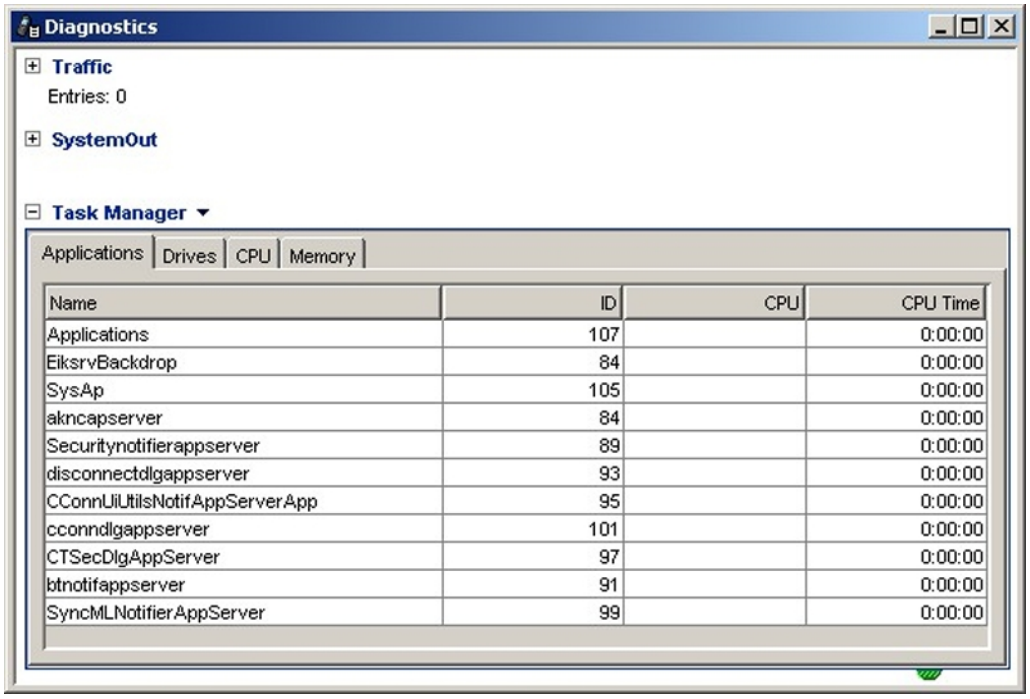


Figure 38: Task Manager pane full view - Applications tab

The full mode includes the four tabs detailed in the following table:

Menu items	Description
Applications	Displays running applications and their process IDs.
Drives	Displays information relating to emulator or device drives. The lower part of the sub-tab provides a slider that adjusts task manager information update speed.
CPU	Displays CPU usage and history.
Memory	Displays RAM and ROM usage.

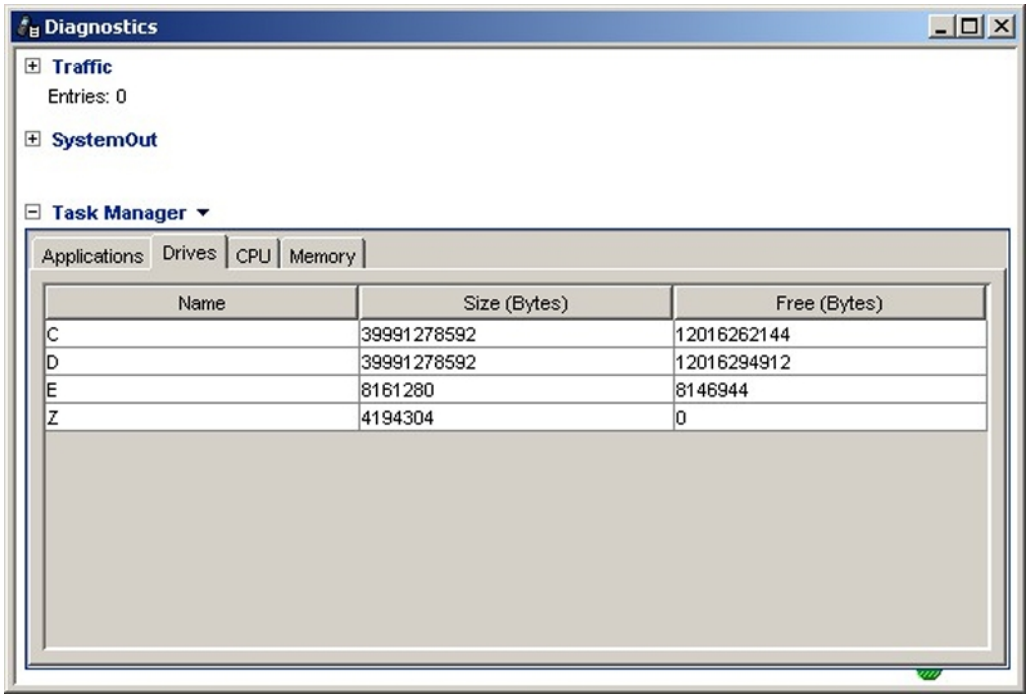


Figure 39: Task Manager pane - Drives tab

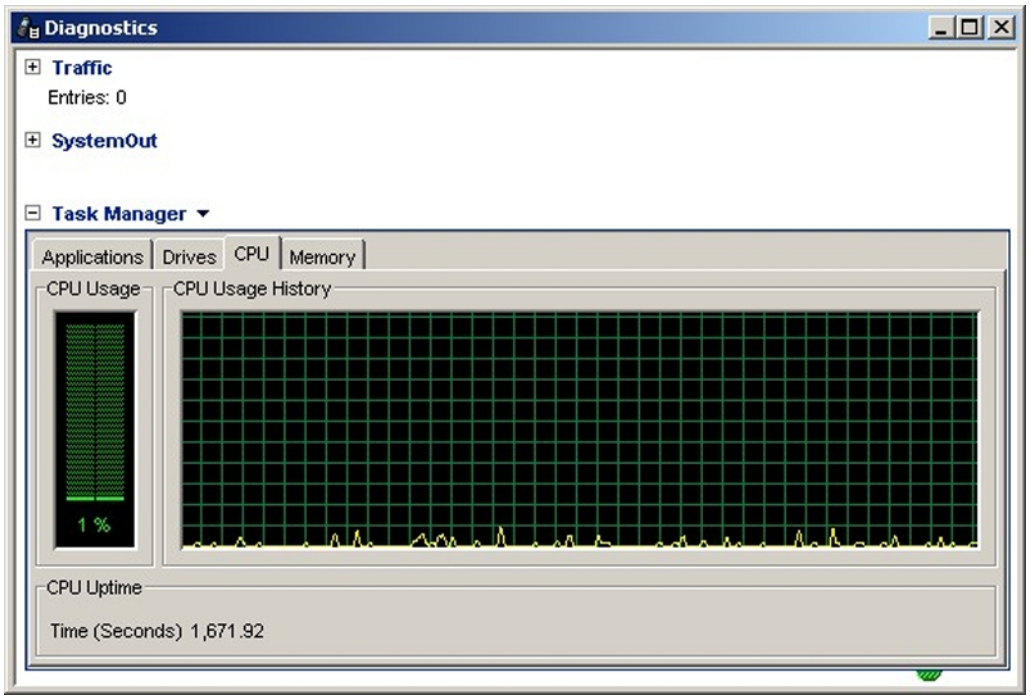


Figure 40: Task Manager pane - CPU tab

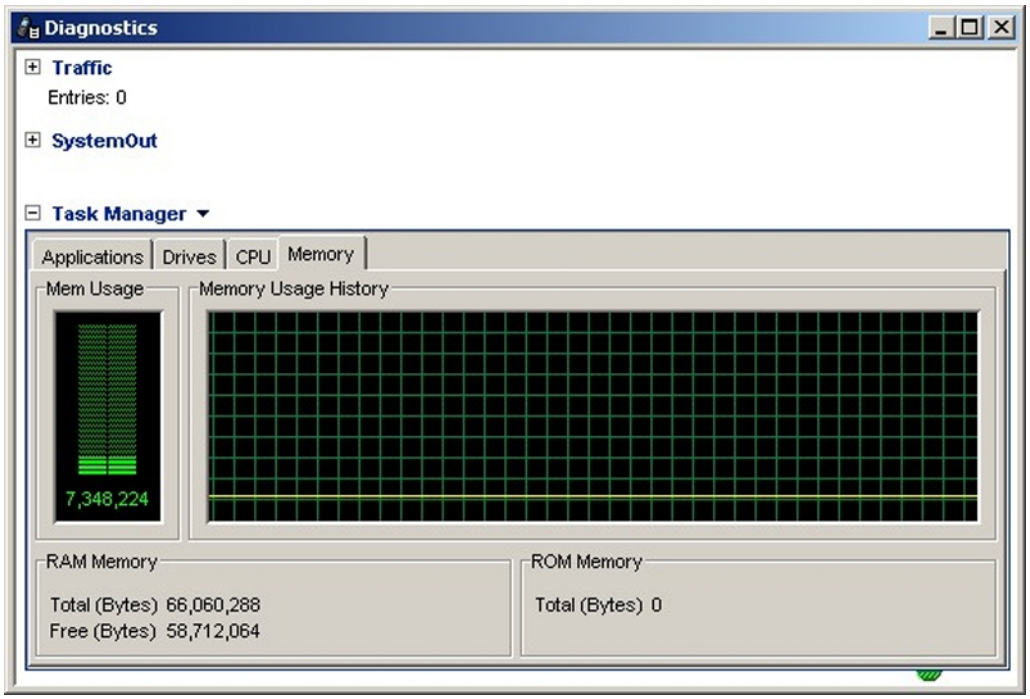


Figure 41: Task Manager pane - Memory tab

3.1.2.7 Closing the emulator

You can close the emulator in the following ways:

- click the 'Close' icon in the Emulator title bar
- select the **File > Exit** from the Emulator menu
- press **Alt+F4**.

3.1.3 About the Emulator

3.1.3.1 Graphical user interface

The S60 emulator Graphical User Interface (GUI) includes a screen and various keys to simulate the function of a target device as closely as possible. To simulate pressing buttons on a real device, click the corresponding keys on the emulator with the mouse.

The emulator screen is divided into three panes: the status pane, the main pane and the control pane. Below the screen you can find the input keys: soft keys, five-way navigation key, start and end call keys and the twelve-way keypad. See the figures below for reference.

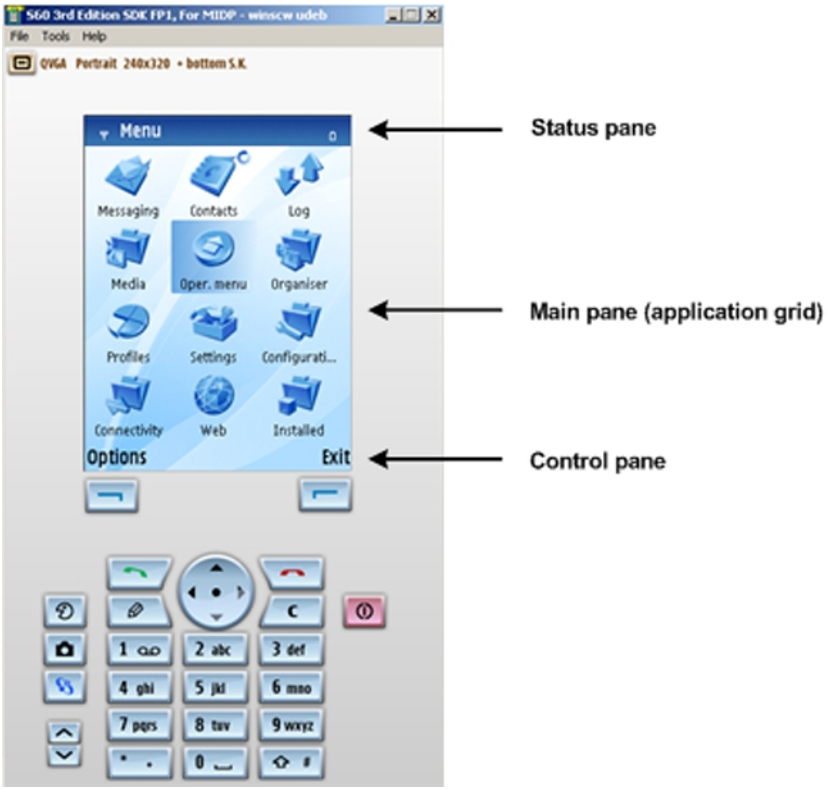


Figure 42: S60 emulator panes

The **Status pane** includes the solid bar near the top of the screen and the area above it. It displays information on the running application and state as well as information on device status, such as signal strength.

The **Main pane** is the principal area of the screen and used for displaying application data.

The **Control pane** is located under the main pane and displays commands and other items related to the soft keys and the navigation key.

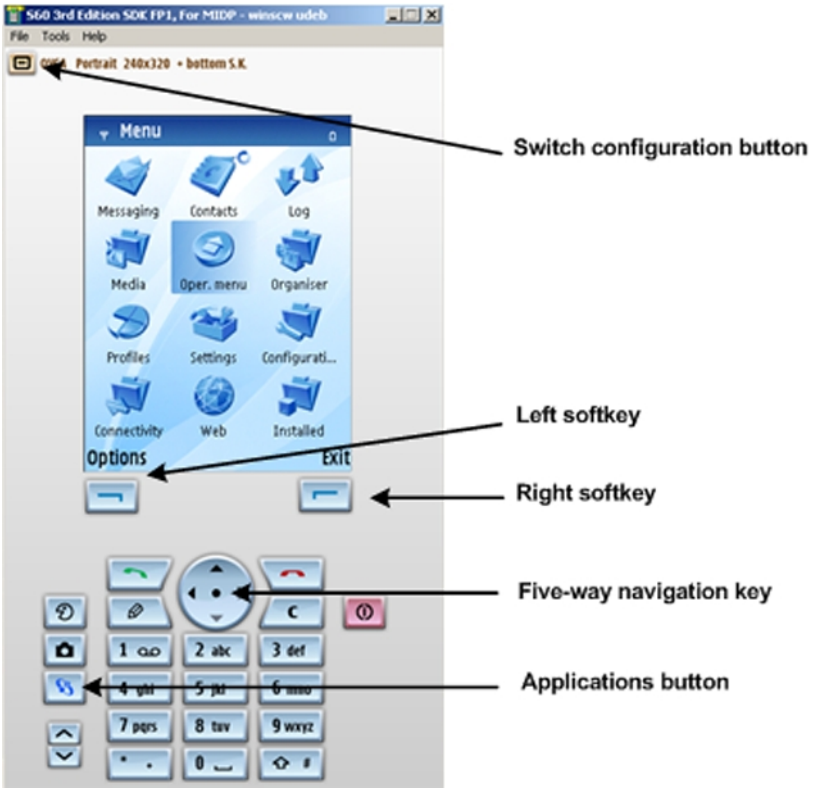


Figure 43: S60 emulator keys and buttons

Use the right and left soft keys to select the options displayed in the control pane. The five-way navigation key provides basic navigation functions for scrolling up, down, left and right on the display.

You can launch applications by clicking the center of the navigation key or clicking **Select** on the left soft key. Exiting applications happens typically by clicking the right soft key. Selecting the **Applications** button takes you to the emulator's application grid (main pane).

For entering data, you can use either the twelve-way keypad or the keyboard.

For details on the using the keyboard and keyboard shortcuts, see Using the keyboard.

3.1.3.2 Tools, menus and dialogs

The S60 emulator graphical user interface (GUI) provides the following tools, menus and dialogs for using the emulator and accessing support documentation.

The following menu items and dialogs are available from the main menu:

Main menu item	Menu item	Action
File	Open...	Enables you to open file in the emulator (see Opening files in the emulator).
	Open URL...	Enables you to open a URL and view it in the emulator (see Opening a web page in the emulator).
	Exit	Closes the emulator.
Tools	Switch Configuration	Enables you to change the emulator resolution (see Changing the emulator resolution).

	Diagnostics	Displays the Diagnostics window which displays debugging information about the applications running in the emulator.
	Utilities	Displays the Utilities window which provides an interface for event simulation.
	Preferences	Displays the Preferences window which is used for configuring global emulator settings and connections.
Help	Release Notes	Launches the Release Notes.
	User's Guide	Launches the SDK Help.
	About...	Provides SDK and platform version information.

3.1.3.3 Diagnostics window

The Diagnostics window displays diagnostics data from the emulator or a device. The tabs of this window can be minimized or maximized by clicking the **+** or **–** buttons on the left of the tab titles.

You can open the **Diagnostics** window from the emulator by selecting **Tools > Diagnostics** from the menu bar.

Tab-specific menus can be opened by moving the mouse on top of a tab and right-clicking the mouse. The size of each pane can be adjusted through a pane height modifier (a green semi-circle) at the bottom of each pane.

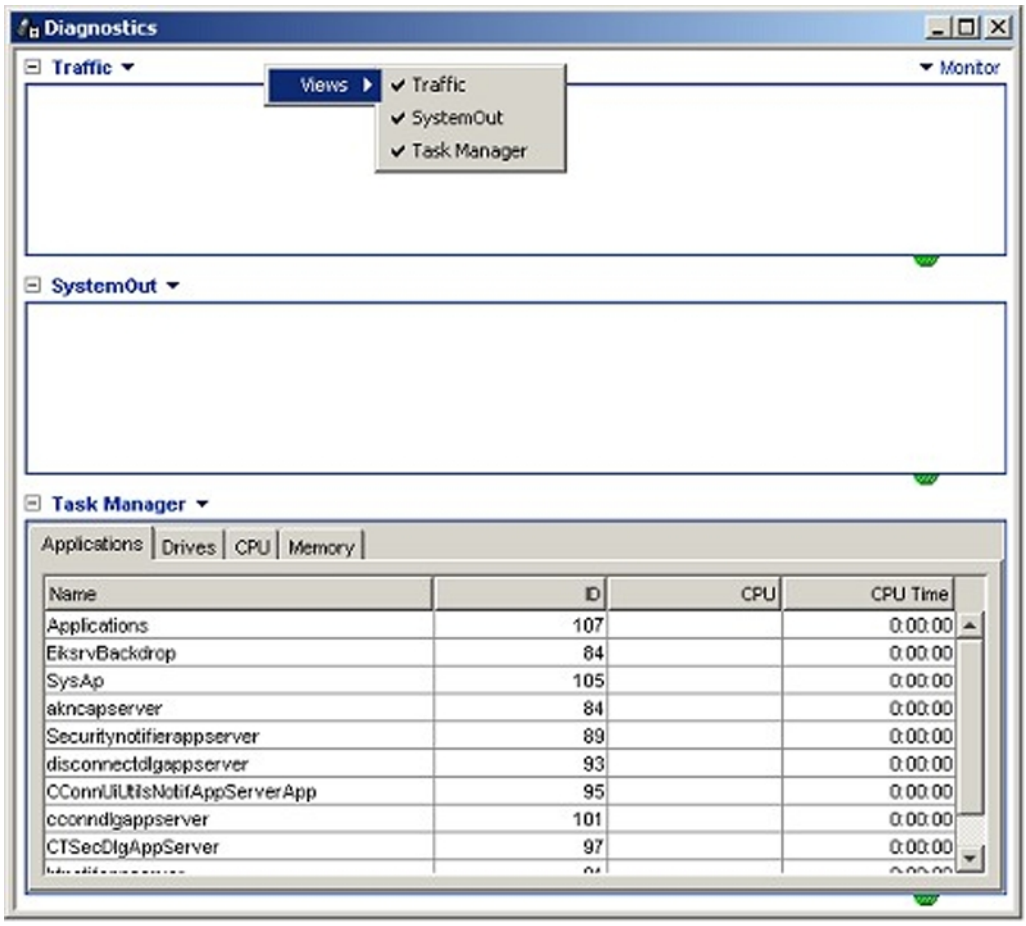


Figure 44: Diagnostics window

The **Diagnostics** window provides panes that display different diagnostics type data from the emulator or a device. The functionality of the **Diagnostics** window and its panes is explained in more detail in the Diagnostics and Tracing section.

Please refer to Diagnostics window panes for more information on how to use the Diagnostics window for diagnostics and tracing purposes.

3.1.3.4 Utilities window

The **Utilities** window contains tool programs that implement runtime emulator related functions. It provides a user interface for:

- simulating various different kinds of phone events through the **Events** tab
- developing location-based applications with the SimPSY plug-in through the **Route** tab.

To open the Utilities window, select **Tools > Utilities** from the emulator menu bar:

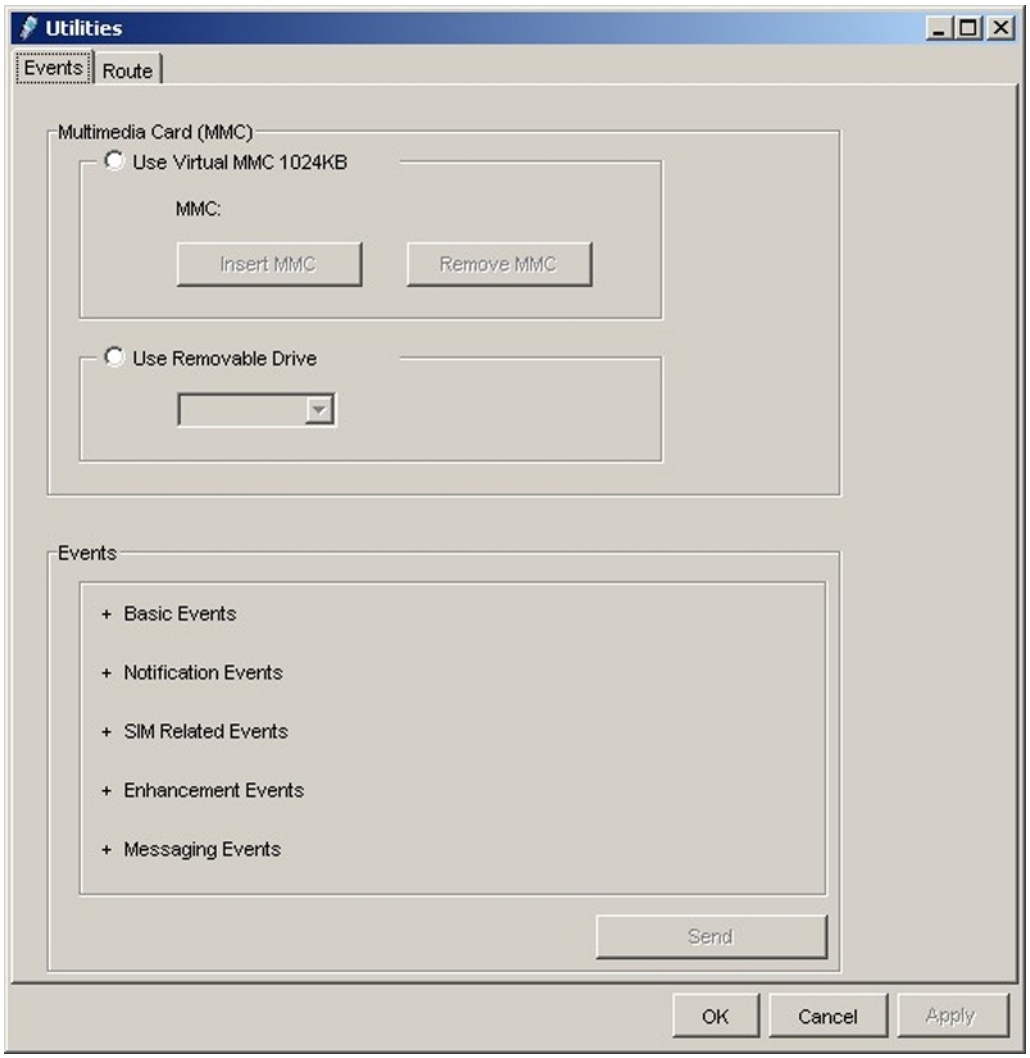


Figure 45: Utilities window

For instructions on how to utilize the features provided through the **Utilities** window, please refer to:

- Events tab
- Route tab - see SimPSY Route dialog.
- Using the Route tab - see Simulation PSY Route Plug-In User's Guide.

3.1.3.5 Preferences window

3.1.3.5.1 Preferences window

The **Preferences** window provides an easy-to-use user interface for configuring global emulator settings, the emulator language and network connectivity.

To open the **Preferences** window, select **Tools > Preferences** from the emulator menu bar.

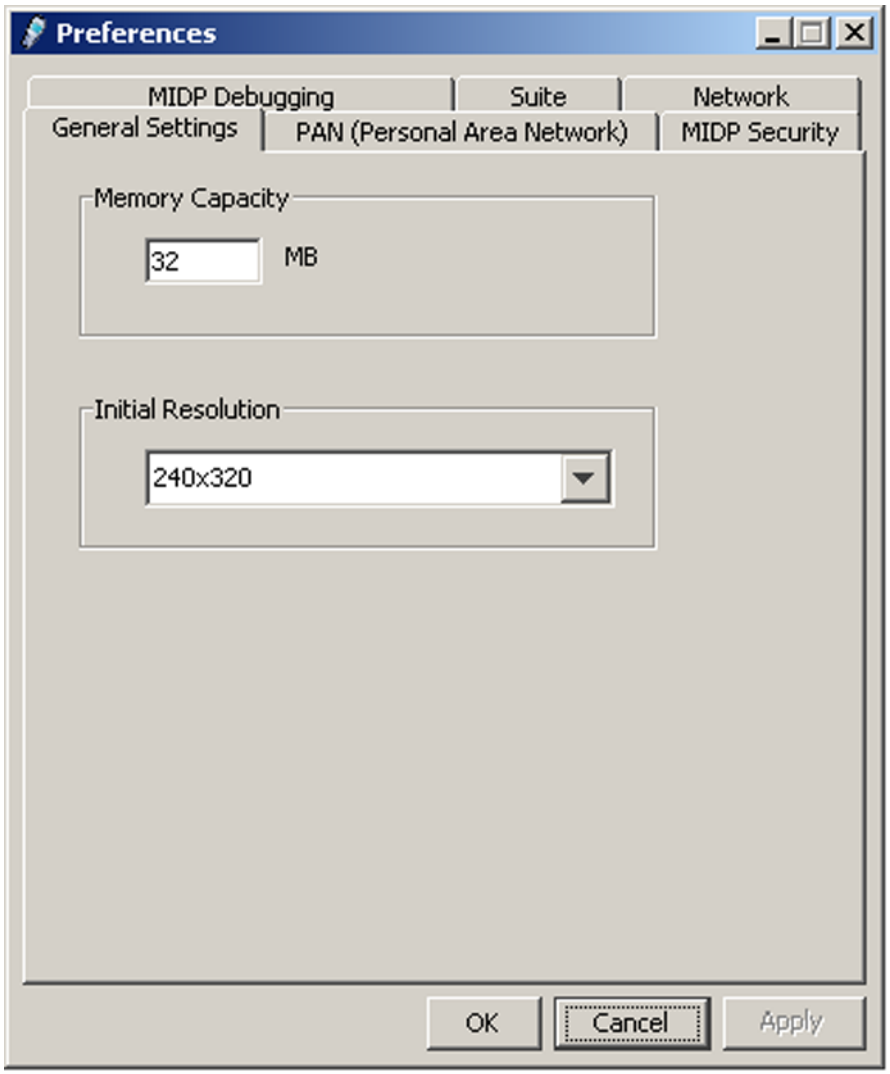


Figure 46: Preferences window

The **Preferences** window provides the following tabs:

- General Settings
- PAN (Personal Area Networking)
- MIDP Security
- MIDP Debugging
- Suite
- Network

3.1.3.5.2 General Settings tab

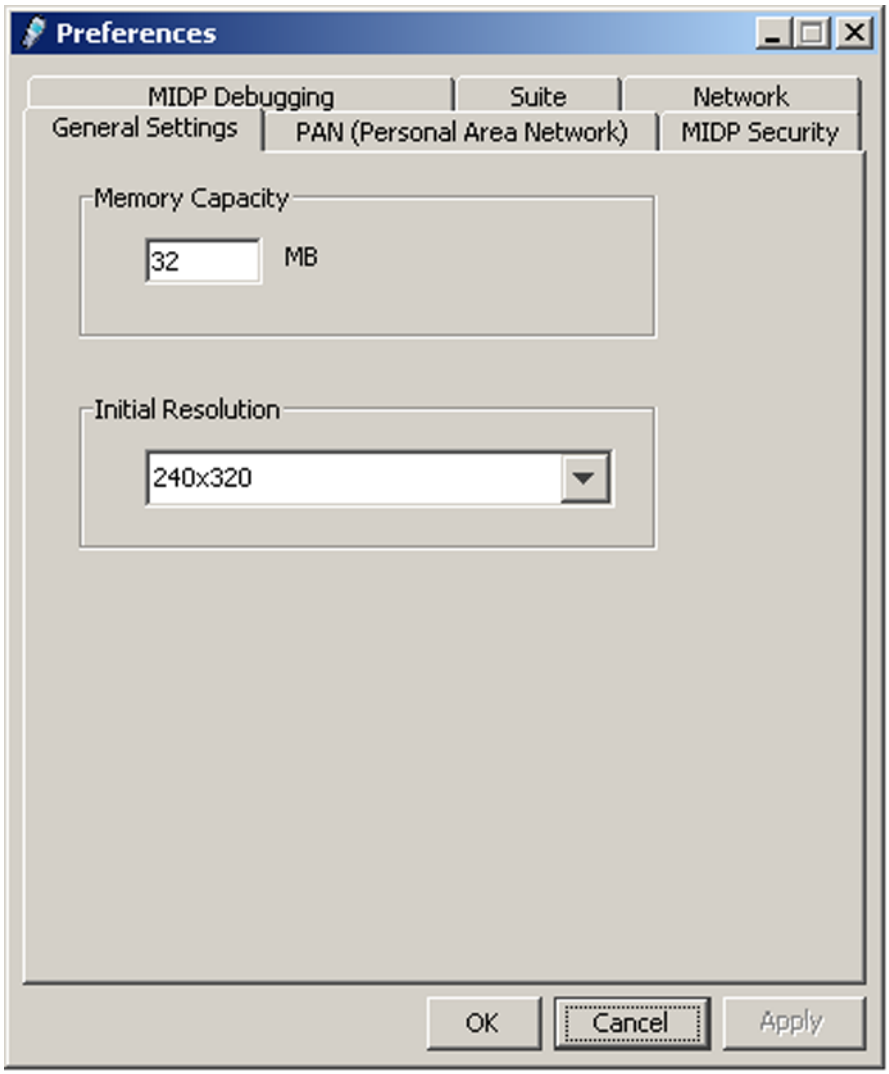


Figure 47: General Settings tab

The **General Settings** tab includes the following options for modifying global emulator settings.

- **Emulator memory capacity**

  Sets the maximum total heap size in the initialization file `epoc.ini`.

  The valid range is 8-256 MB. This setting allows you to emulate target devices with different memory capacities (maximum total heap size in the initialisation file `epoc.ini`).

	Note: This option requires emulator restart to activate the changed settings.

- **Initial resolution**

  The options are:

  - 240x320
  - 320x240

These settings are activated after emulator restart.

3.1.3.5.3 PAN (Personal Area Networking) tab

The **PAN** tab provides options which are used for configuring the Bluetooth and Infrared connections for the emulator. By setting these options you can enable Bluetooth and IrDA connections to the emulator.

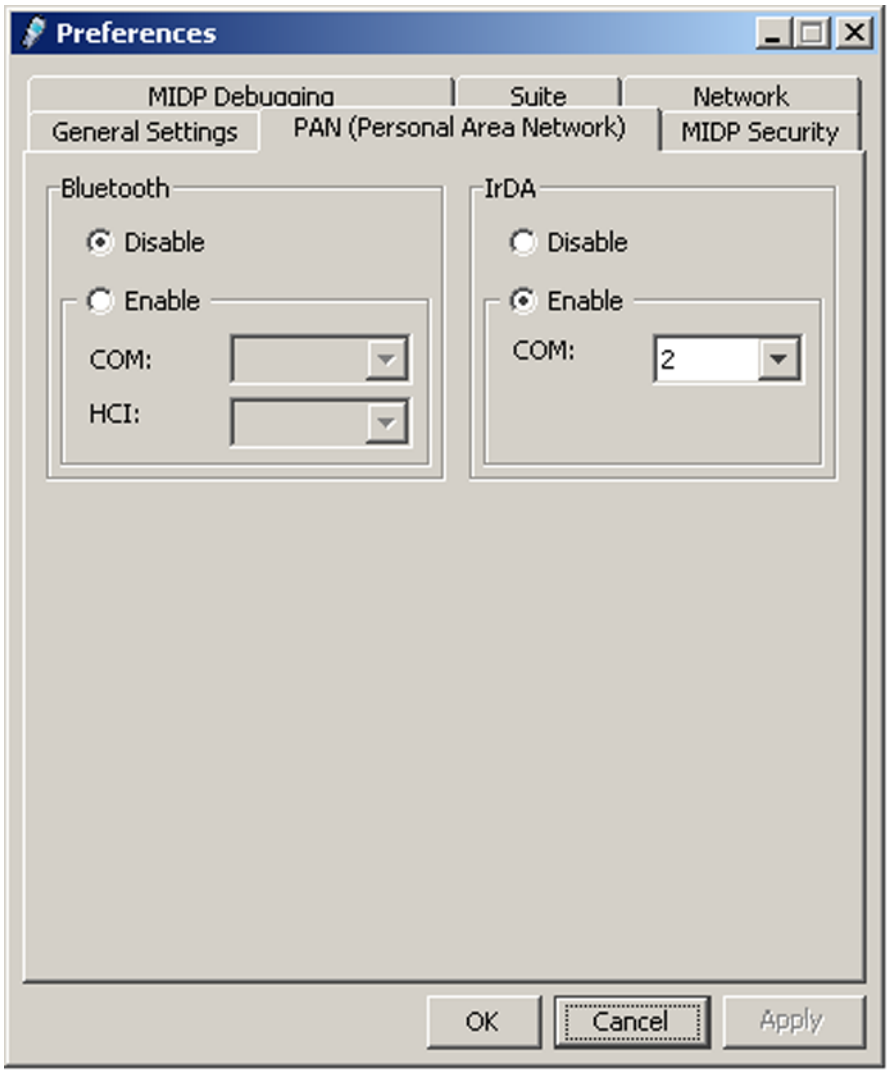


Figure 48: PAN (Personal Area Network) tab

For configuring a Bluetooth connection:

Option field	Description
Bluetooth	Sets the used bluetooth connectivity configuration. **COM:** Physical com port used by the Bluetooth card. Sets the hci setting in the `\Epoc32\Winscw\c\System\Data\bt.esk` file. **HCI:** Bluetooth stack (BCSP or H4) Sets the protocol stack that the emulator uses to communicate with Bluetooth hardware. The default is the BCSP protocol stack.

IrDA	Sets the used infrared connectivity configuration. Physical com port used by the Ir Pod. The emulator uses serial communications to interface directly with the PC's hardware, which means that an Ir Pod connected to the COM port should be used.

The settings are activated after emulator restart.

For more detailed configuration instructions, see Configuring Bluetooth.

For a list of possible error messages displayed while using the Preferences window, please refer to Troubleshooting.

For detailed information about the security domains and their use, see the document Recommended Security Policy for GSM/UMTS Compliant Devices, an addendum to the Mobile Information Device Profile (MIDP) version 2.0 for the Java™ 2 Platform, Micro Edition (J2ME™) specification. You can download the specification from the Java Community Process website at http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html.

3.1.3.5.4 MIDP Security tab

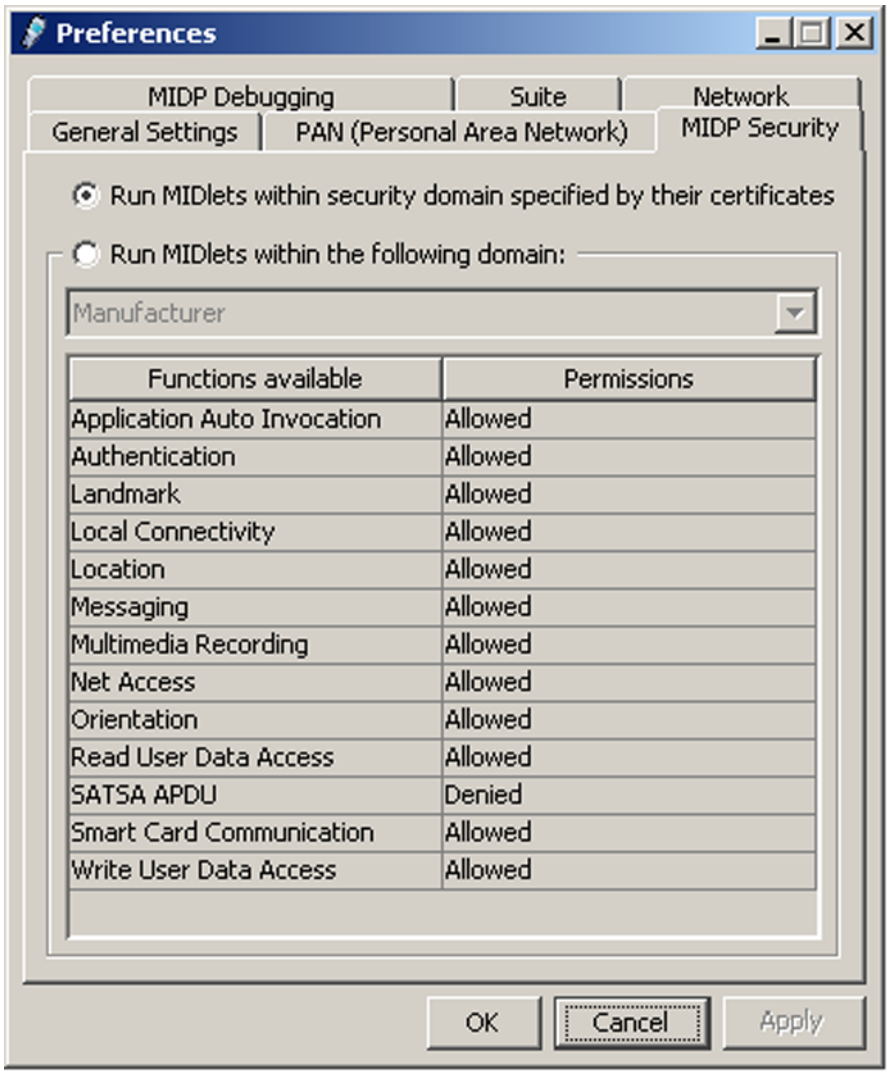


Figure 49: MIDP Security tab

The **MIDP Security** tab is used for adjusting MIDlet runtime security sandbox on the emulator. The tab has two main options:

- **Run MIDlets within security domain specified by their certificates** (default)

  Selects security environment according to the real domain and provided certificates.
- **Run MIDlets within the following domain**

  Lets the user specify the desired domain from the list or specify available function permissions manually in more detail (with user defined “domain”). The defined domains are:

  - Manufacturer
  - Operator
  - Trusted Third Party
  - Untrusted
  - User Defined.

Domains are predefined sets of function permission mappings. The domain User defined is the only domain that allows permission changing. When User defined domain is chosen, permission cells become editable. By clicking the **Permissions** cell, the selectable permission list becomes visible and the user can define permissions for each function.

Permission	**Description**
Allowed	API access is allowed.
Denied	API access is denied.
Ask once	When accessing to API the first time, the user is asked to confirm the action.
Ask every time	Confirmation is done every time accessing API.

The configurations require emulator restart to be activated.

3.1.3.5.5 MIDP Debugging tab

The **MIDP Debugging** tab consists of the following three panes:

- S60Emulator
- S60Device_over_Bluetooth
- S60Device_over_wlan

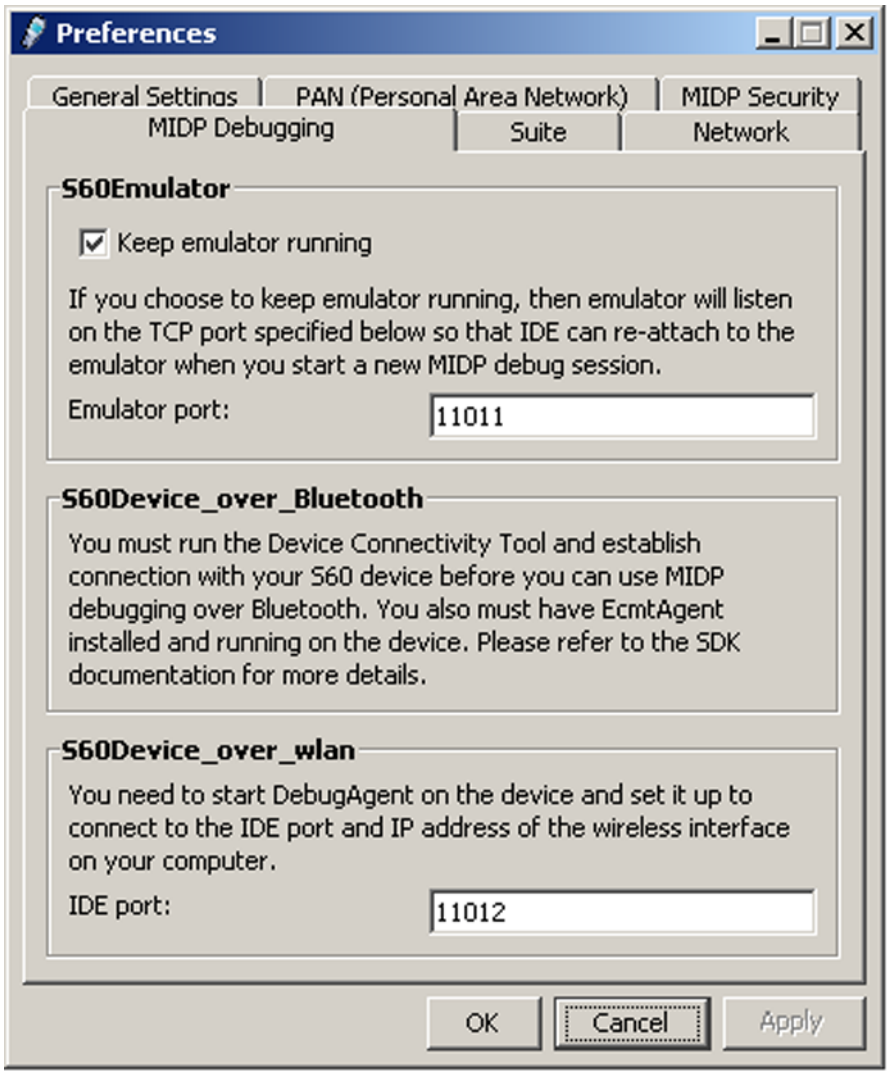


Figure 50: MIDP Debugging tab

S60Emulator

If you want to choose to keep the emulator running, select the **Keep emulator running** check box.

By choosing this option, the emulator listens on the TCP port specified in the **Emulator port** field. The IDE can re-attach to the emulator when you start a new MIDP debug session.

S60Device_over_Bluetooth

Before using MIDP debugging over Bluetooth, proceed as follows:

- Run the Device Connectivity Tool and establish a connection with your S60 device.
- Check that you have the EcmtAgent installed and running on the device.

S60Device_over_wlan

Proceed as follows:

1 Start the DebugAgent on the device.

2 Set it up to connect to the IDE port and IP address of the wireless interface on your computer.

3.1.3.5.6 Suite tab

Suite tab is used for configuring MIDlet packaging specifics. The applied values are used when the emulator packages MIDlet and resources into `.jad` and `.jar` files.

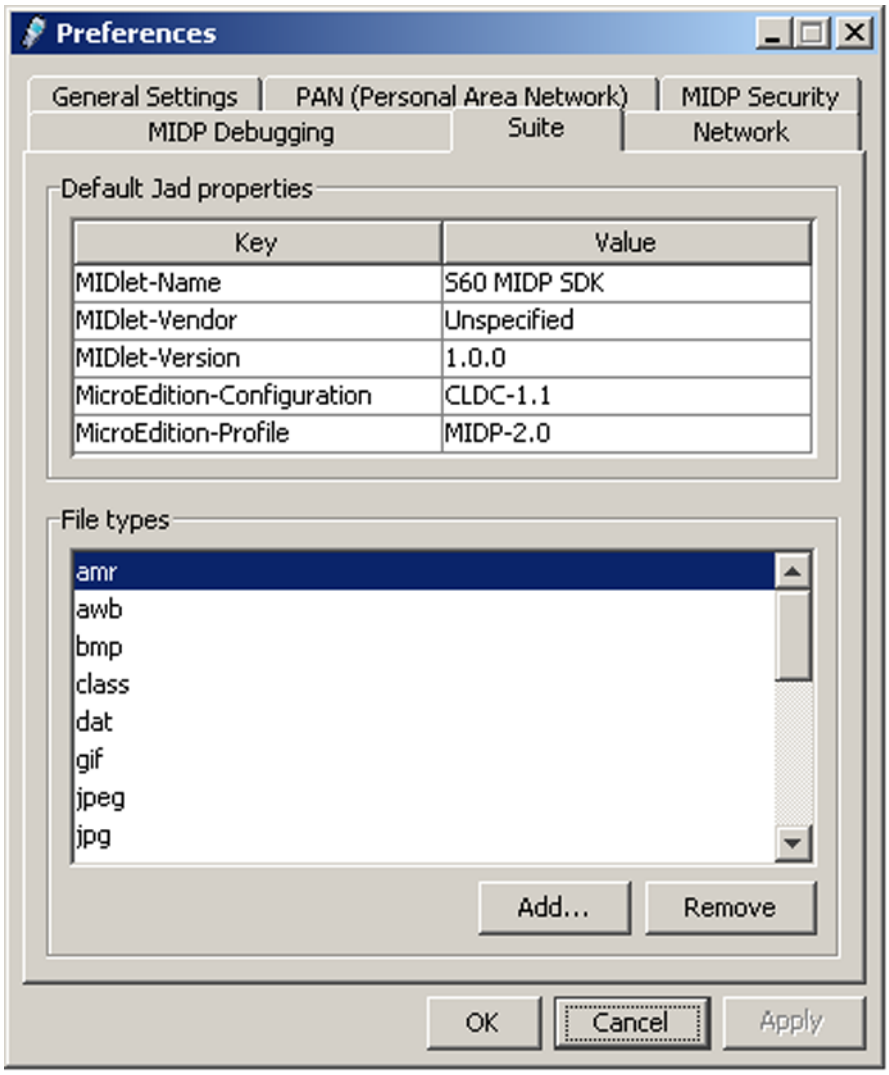


Figure 51: Suite tab

The **Suite** tab has two option fields, **Default Jad properties** and **File types**. They provide the following options:

Option field	Description
Default Jad properties	The table allows the user to define `.jad` file properties to be used included in the generated `.jar` file.
File types	File type extension list that enables the user to add and remove file types.

3.1.3.5.7 Network tab

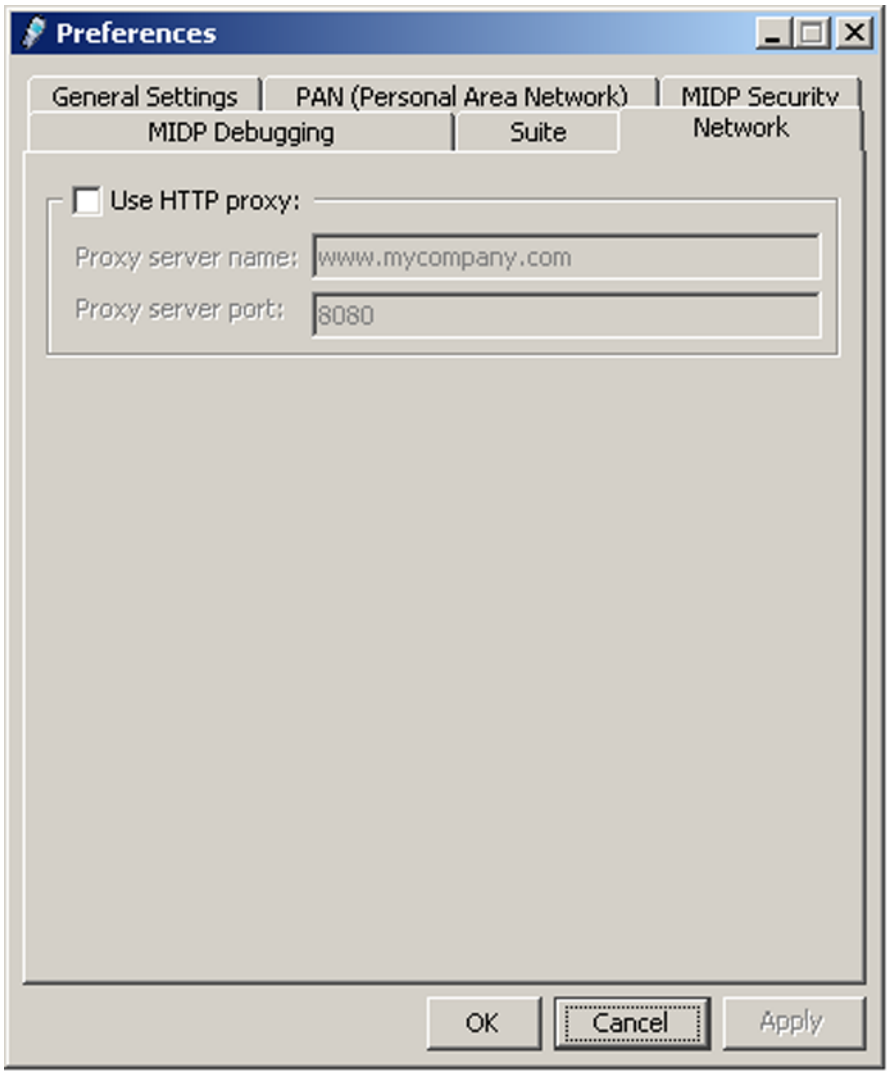


Figure 52: Network tab

The **Network** tab is used for setting up emulator network settings. It offers the following http proxy setting options:

Option	Description
Use HTTP proxy	Used for defining whether to take HTTP proxy into use or not.
Proxy server name	Used for setting proxy server name
Proxy server port	Used for setting proxy server port

Restart the emulator for the changes to take effect.

3.1.3.6 Improved emulator debugging

In order to speed up the development cycle when debugging with the emulator from an IDE, you do not need to restart the emulator for every debug session. You can leave the emulator running, rebuild the debugged application from the IDE and launch the application from the **Installed** folder in the emulator.

Note: In some cases you have to close the emulator before rebuilding your application. One such case is when you compile the resource files: In this case the compilation will fail if the emulator is running, as the emulator locks resource files.

3.1.3.7 Scalable UI

Applications implemented for S60 3rd Edition and later should scale to all of the supported resolutions. From this point on, S60 devices may use one of the following screen resolutions:

- QVGA Portrait (240x320)
- QVGA Landscape (320x240)

The scalability requires developers to take different screen resolutions into account when creating their application designs. The emulator supports this by providing the needed resolutions.

Related information

- Changing the emulator resolution
- Setting the default resolution of the Emulator
- Preferences window

3.1.4 Emulator Connectivity

3.1.4.1 TCP/IP support

3.1.4.1.1 TCP/IP support for S60 SDK

The TCP/IP support for S60 SDK is a developer’s tool that lets you access IP networks via for example an Ethernet card. This is useful if your application needs access to the Internet.

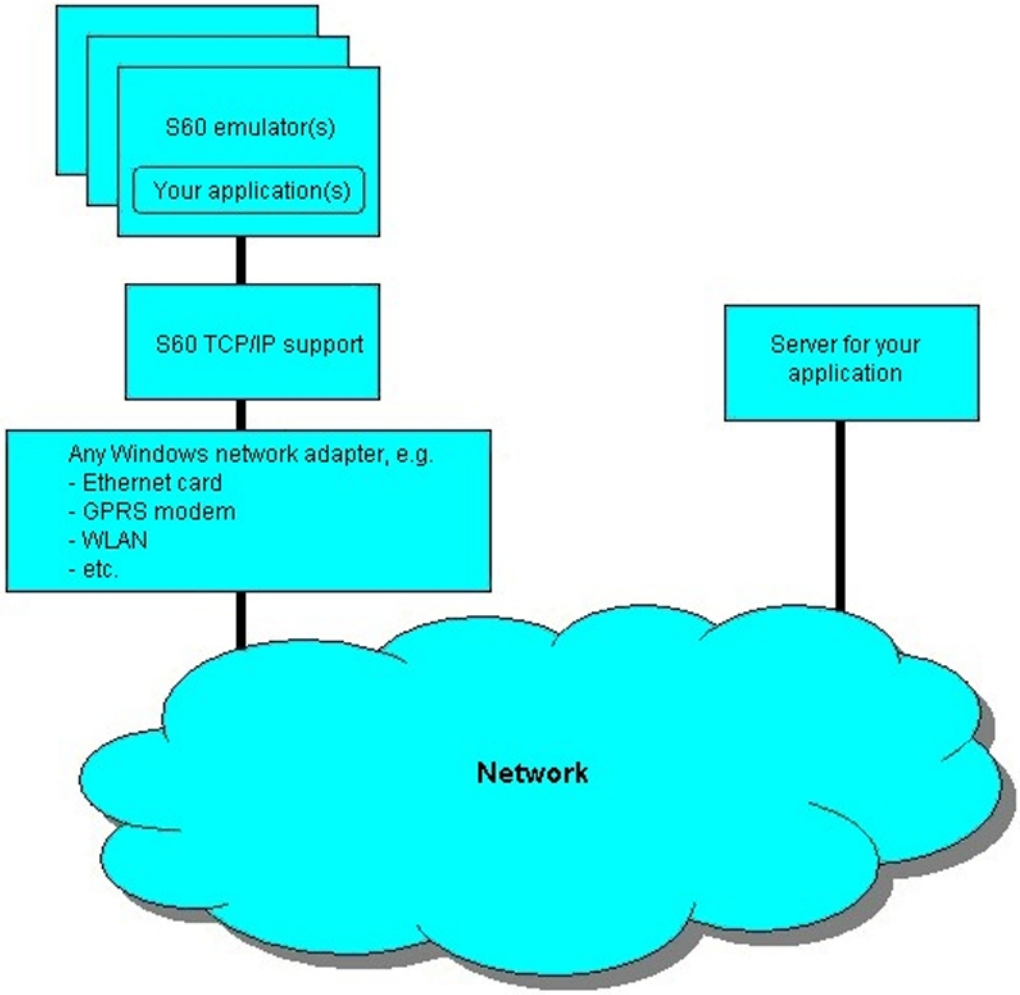


Figure 53: TCP/IP support

When using this WinSock access point the emulator uses the Windows TCP/IP stack, and not the Symbian TCP/IP stack. The emulator (and the applications running in it) use the same IP address as the IP address of the PC. The localhost (127.0.0.1) for Symbian applications are mapped to the localhost of the PC. As a consequence one cannot have a Symbian application and a regular PC application listening on the same local port on the same PC.

As the emulator uses the Windows TCP/IP stack, you can use any Windows networking tool to analyze the behavior of the emulator (for example Ethereal).

If you want to run the emulator on a mobile network (GPRS , WCDMA) you can either use a cellular modem PC card or a mobile device as a modem for connectivity.

The TCP/IP support does not usually affect your application design. Simply use normal means for accessing Sockets in your Symbian OS application. Your application design is affected only if your application is interested in the type of access point used.

The emulator can be used in an IPv6 network environment, if the hosting operation system supports IPv6. Please consult the documentation of the host operation system to install and enable IPv6 on your PC.

3.1.4.1.2 Configuring TCP/IP support

Prerequisites

Please note the following prerequisites in order to be able to make a successful network connection with the S60 SDK:

- You have a PC with Windows™ 2000 or Windows™ XP operating system.

- A working ethernet network connection is in place.
- S60 SDK has been successfully installed on your PC.
- The emulator is up and running. For instruction on launching the emulator, please refer to the Starting the Emulator section.

The HTTP proxy can be selected in the Network tab of the Preferences window

Steps

1. Open the Network tab of the Preferences window.
2. Select the **Use HTTP proxy** checkbox and enter the needed information in the **Proxy server name** and **Proxy server port** fields.

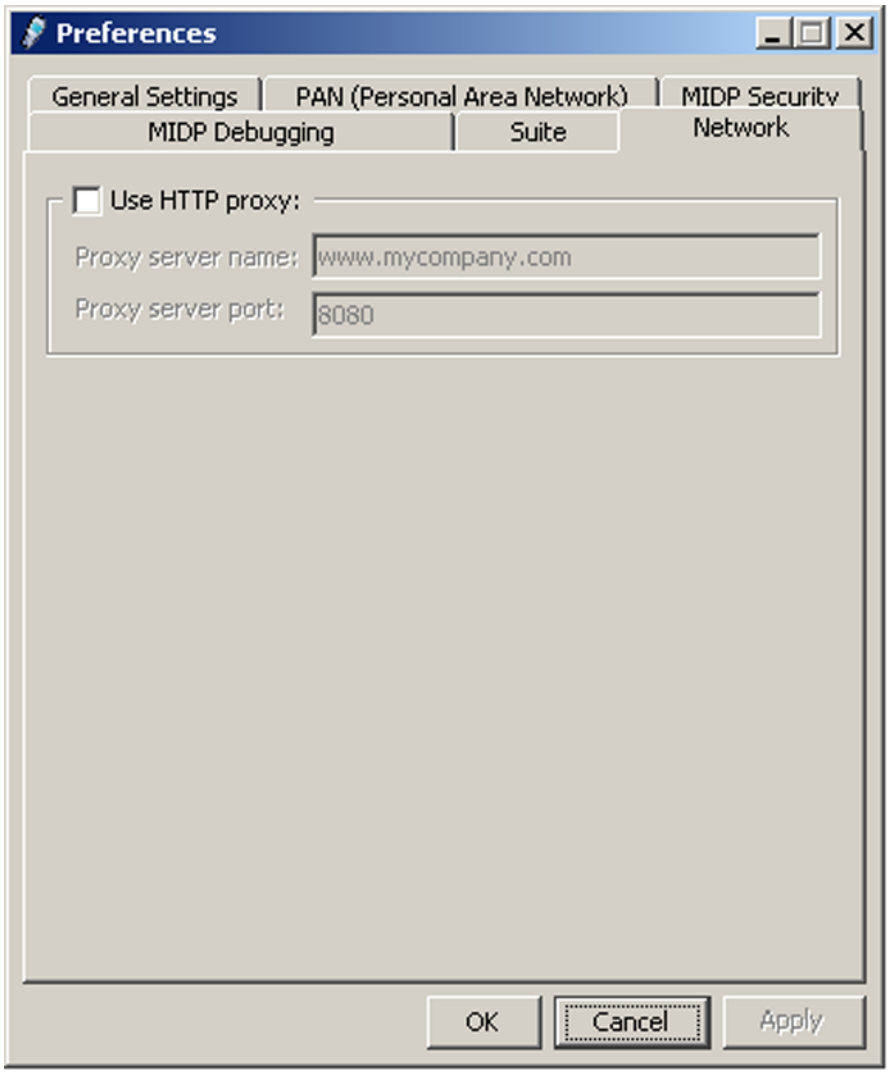


Figure 54: HTTP proxy settings

	Note: Your application has to support connections via proxy. These proxy settings constitute simply a centralized storage facility for this information.

3. Click **OK** to accept the changes.
4. Exit the Preferences window.

Results

The Ethernet connection is now configured for the emulator using the specified HTTP proxy. To test that the connection works, follow instructions provided in Using TCP/IP Support with the emulator.

3.1.4.1.3 Using TCP/IP support with the emulator

Steps

1 Start the emulator.
.

2 In the emulator , open the application grid by clicking the Applications button (see
. Graphical user interface)

3 In the application grid, open the **Web** application.
.

Figure 55: Web application

4 Enter the URL of a web server in your network in the **Address** field.
.

5 Select **Go to**.
.

6 Select the access point.
.

Results

If your configuration was successful, the web page in question opens in the emulator and TCP/IP support for the emulator is ready for use.

3.1.4.2 Bluetooth support

3.1.4.2.1 Configuring Bluetooth

Prerequisites

Before you start, make sure that:

- the emulator is up and running.
- the Bluetooth card is connected on your PC. For the supported Bluetooth cards, see section Supported Interfaces, tool set-ups and accessories.
- Bluetooth service is not used by any other Windows application.

Steps

1. Select **Tools > Preferences** from the emulator main menu to access the Preferences window,
2. Open the **PAN (Personal Area Networking)** tab.

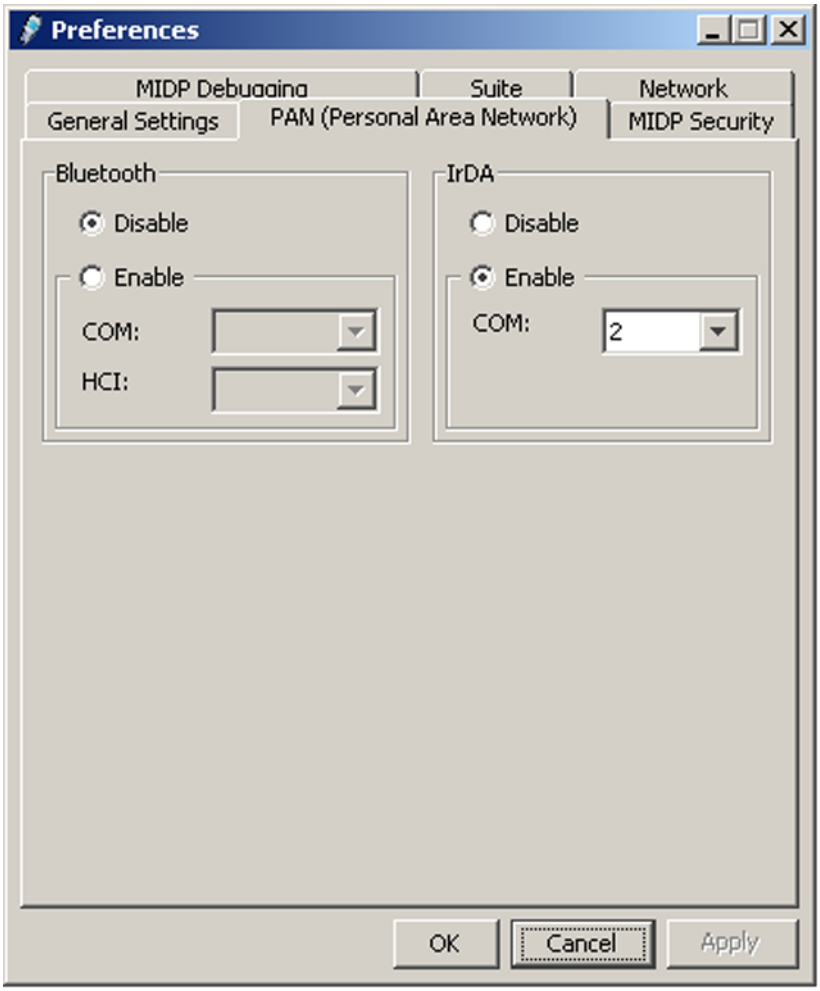


Figure 56: PAN (Personal Area Networking) tab

3. Select **Enable** under **Bluetooth** and enter the following information for the connection:
   - Select the COM port used by the Bluetooth card.

     For devices using RS-232 connection (for example, Casira), select the physical COM port of your PC, otherwise use the virtual COM port for the installed device. To access the virtual COM port information, go to **Control Panel > System** and select **Hardware > Device Manager > Ports**.
   - Select the Bluetooth Host Configuration Protocol (HCI) (BCSP or H4).

     The used HCI protocol depends on the Bluetooth hardware. If you are unsure about which protocol your device supports, please contact the Bluetooth card manufacturer. Casira, for instance, supports both interface protocols.

     By default, the emulator uses the BCSP HCI to communicate with Bluetooth hardware.
4. Click **Apply** to accept the changes or **OK** to accept and exit the tool.

   A notification message is displayed stating that you need to restart the emulator to activate the Bluetooth connection.

Results

You can now proceed with enabling Bluetooth for the emulator to test that the connection works.

3.1.4.2.2 Enabling Bluetooth

Context

To enable Bluetooth either on an emulator, or on a target device, do the following:

Steps

1. Go to the application grid of the device.

Figure 57: Application grid

2 In the application grid, select and open the **Connectivity** folder.

. The following connectivity options are displayed:

Figure 58: Connectivity options

3 Select and open **Bluetooth**.

. The following **Bluetooth setup** screen appears:

Figure 59: Bluetooth setup screen

4 In the **Bluetooth setup** dialog, do the following:

.

- Enter a name for your phone. This is the name that other devices will be seen when they discover new Bluetooth devices.
- Set **Bluetooth** to **On** by either pressing the selection key or by selecting **Options > Change** with the left softkey. This might take a few seconds.
- Set **My phone's visibility** to **Shown to all**. This enables other Bluetooth devices to always see your device when Bluetooth is activated.

Results

Bluetooth is now enabled on your device, and you can test the connection, for example, by sending a notepad message to another device that has Bluetooth enabled.

Examples

You can also test the connection by performing a pairing with a remote device as follows:

1 On the **Bluetooth** screen, navigate to the right on the status pane to access the **Paired devices** tab and select **New paired device**.

All Bluetooth devices that are in the range are displayed in a list. Note that devices which are set as 'Hidden' are not displayed.

2 Press **Select** when you are on the device you want to pair with.

3 You may be requested to enter a passkey to complete pairing. Enter the appropriate key for both devices (a single digit is enough).

4 The next dialog asks if the device should be authorized to make connections automatically. Answer 'Yes' so that future connections between the two devices can be established without any confirmation screens.

5 When the pairing is completed successfully, an information dialog is displayed.

3.1.4.3 Infrared support

This section describes the configuration steps that you need to carry out in order to enable Infrared connectivity for the emulator manually. The emulator uses serial communications to interface directly with the PC's hardware, which means that an Ir Pod connected to the COM port should be used.

There is no need to install any Windows drivers for the Ir pod when it is used with the emulator.

The Symbian OS IrDA stack supports the RS-232 Extended Systems, Inc. Jeteye pods ESI 9680-7201, ESI 9680-7401, and ESI 9680-7501. Other pods are not guaranteed to work properly.

Prerequisites

Before you start, make sure that:

- the emulator is up and running
- you have a supported type of Ir pod connected to the COM port.

Configuring Infrared connections

Infrared connections are configured using the Preferences window.

1 To access the **Preferences** window, select **Tools > Preferences** from the emulator main menu.

2 Click the **PAN (Personal Area Networking)** tab to display the following dialog:

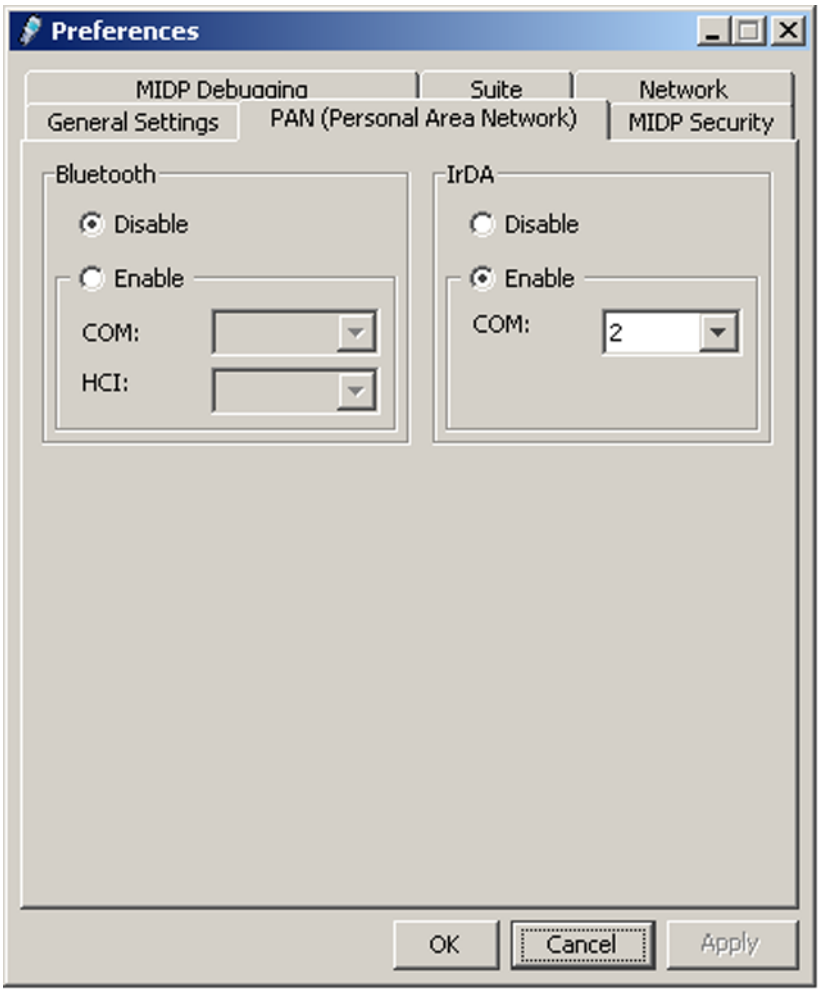


Figure 60: PAN (Personal Area Networking) tab

3 Select **Enable** under **IrDA**.

4 Select the virtual COM port used by the Ir pod.

	Note: Make sure that you use different COM ports for the Bluetooth and infrared connections.

5 Click **Apply** to accept the changes or **OK** to accept and exit the tool.

A notification message is displayed stating that you need to restart the emulator to activate the changes.

3.1.4.4 SMS/MMS support

The S60 SDK emulator SMS/MMS support enables 3rd party application to test applications using SMS/MMS APIs in the emulator. This speeds up development of messaging-based applications in a cost-efficient way.

SMS/MMS support in the S60 SDK emulator is based on file based sending and reception:

- **Sending an SMS/MMS:** The emulator writes the sent message to a directory specified for SMS/MMS sending. Any external application interested in the message can read it from the specified directory.

	Note: The message will remain in the directory if not removed.

- **Receiving an SMS/MMS:** To receive an SMS/MMS, move the message to the specified directory for SMS/MMS reception. The emulator reads the message from the specified location and shows the message in the Inbox of the emulator.

	Note: You can also copy example SMS/MMS messages to the Inbox folder of the emulator with the **Messaging events** of the **Utilities > Events**. Please refer to Simulating messaging events.

The table below describes the directories and prefixes used for sending and receiving SMS and MMS messages in the emulator.

Table 3.2: Directories and prefixes used for sending and receiving SMS and MMS

Operation	Directory	Prefix
Send MMS	`<SDK_home>\bin\epoc32\winscw\c\mmsout`	mms
Receive MMS	`<SDK_home>\bin\epoc32\winscw\c\mmsin`	mms
Send SMS	`<SDK_home>\bin\epoc32\winscw\c\smsout`	sms
Receive SMS	`<SDK_home>\bin\epoc32\winscw\c\smsin`	sms

Supported SMS specification is GSM 03.40 version 7.4.0 Release 1998.

Supported MMS specification is MMS WAP-209-MMSEncapsulation Protocol.

3.1.5 Configuring the Emulator

3.1.5.1 Setting the emulator language

Context

The default language of the emulator UI is English. However, the language can be changed to any of the other provided languages. To change the language, do the following:

Steps

1 Close the emulator.

.

2 Select the desired language from the **Start** menu folder of the SDK.

. Select Start > All Programs > S60 Developer Tools > 3rd Edition SDK FP 1 > MIDP > Languages

3 Restart the emulator.

.

Next actions

	Note: The emulator should not be running during the language change. If it is running, the language change will fail. In such a case, you need to close the emulator application and perform the language change again.

3.1.5.2 Setting the default resolution of the emulator

To set the default resolution of the emulator, do the following:

1 Open the Preferences window.

2 Select the **General Settings** tab:

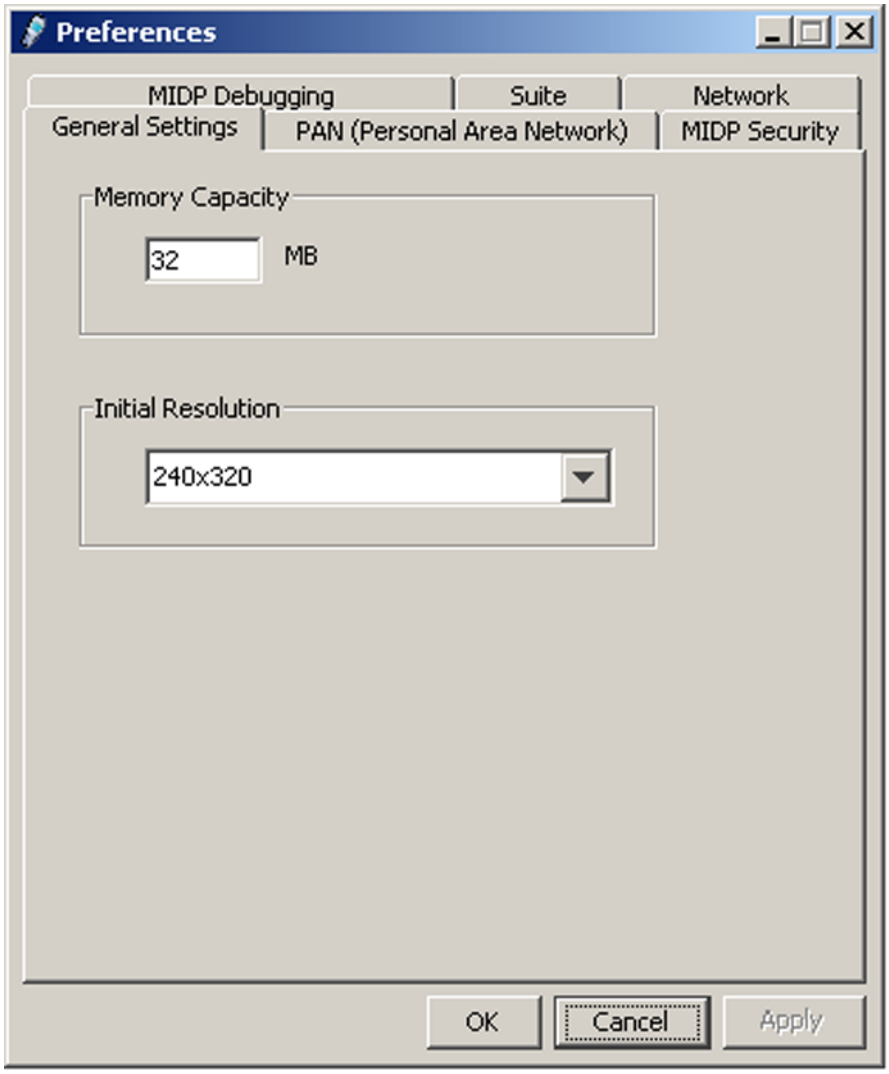


Figure 61: General Settings tab

3 Select the desired resolution from the **Initial Resolution** drop-down menu.

The options are:

- 240x320 (QVGA Portrait)
- 320x240 (QVGA Landscape)

4 Click **OK**.

5 Restart the emulator.

The selected resolution will now be applied in the emulator user interface.

Related information

- Changing the emulator resolution
- Preferences window
- Scalable UI

3.1.5.3 Setting the emulator memory capacity

To set the memory capacity of the emulator, do the following:

1 Open the Preferences window from the main menu by choosing **Tools > Preferences**.

2 Open the **General Settings** tab:

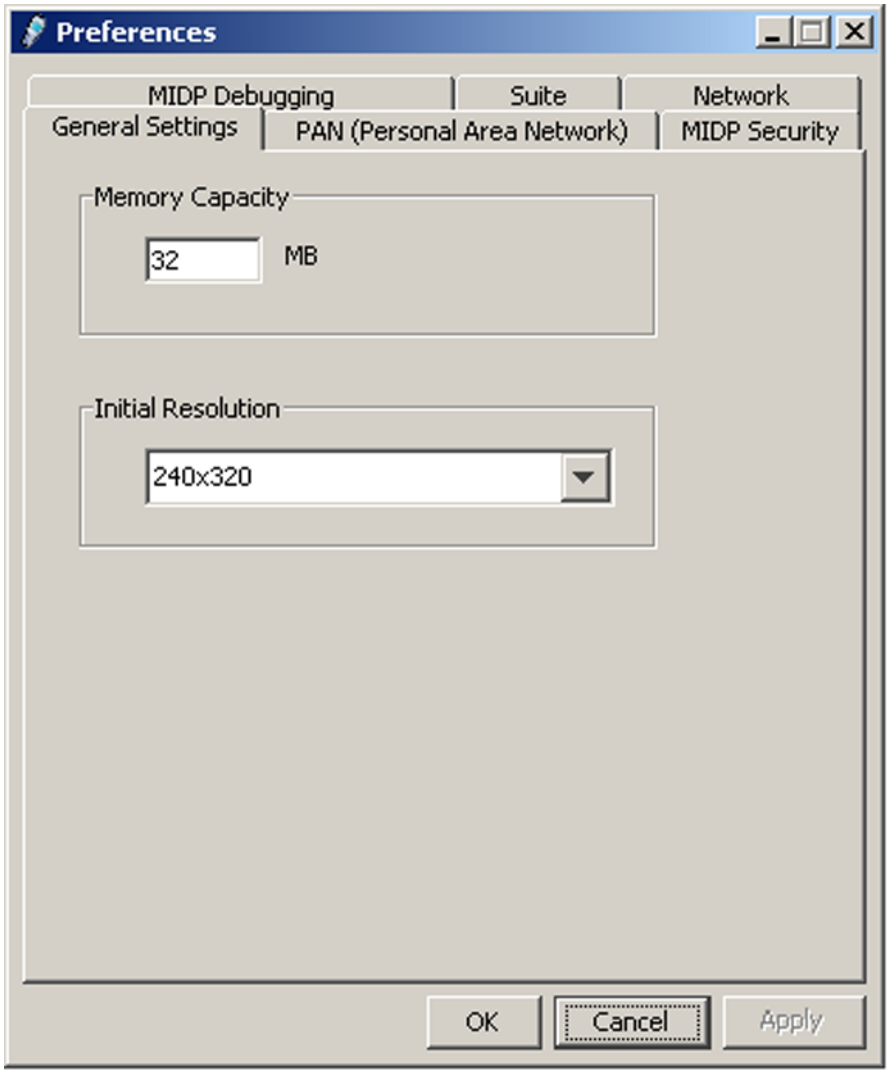


Figure 62: General Settings tab

3 Change the value in the **Memory Capacity** field (maximum total heap size in the initialization file `epoc.ini`) to emulate target devices with different memory capacity. The valid range is from 8 to 256 MB. The default is 32 MB.

4 Click **OK**.

5 Close and restart the emulator.

The changed memory capacity value will now be applied in the emulator.

3.1.6 Troubleshooting

3.1.6.1 Error messages

This section provides information on problem situations and error messages you may encounter while working with the emulator and instructions on how to solve them.

The following table lists error messages related to the Preferences window in alphabetical order.

Message	Symptom	Cause	Action

Cannot select the same COM port for both Bluetooth and IrDA.	Configuring Bluetooth or infrared connectivity fails.	It is not possible to use the same port for the Bluetooth and IrDA connection.	Configure different COM ports for the Bluetooth and infrared connections.
Java.exe not found on system path.\n\n \Java Runtime Environment (JRE) version 1.4 or greater is required.	Starting the Preferences window fails.	You do not have Java Runtime Environment installed on you PC.	To install the required virtual machine: 1 Go to the Source for Java Developers website at http://java.sun.com. 2 On the Source for Java Developers website, locate the J2SE SDK download link and click it. 3 Follow the download and installation instructions.
Please enter a value between 8 - 256.	Configuring the memory capacity fails.	The value entered for memory capacity is not valid.	Enter a value within the specified range: 8-256 MB.

Requirements You will need a Java 1.4.1 or later compatible virtual machine installed on your system.	Starting the Preferences window fails.	Your Java virtual machine version is earlier than the required version.	See the **Details** and **Info** tabs on the Requirements dialog. The **Details** tab shows the version installed on your system. The **Info** tab shows the required version. To install the required virtual machine: 1 Go to the Source for Java Developers website at http://java.sun.com. 2 On the Source for Java Developers website, locate the J2SE SDK download link and click it. 3 Follow the download and installation instructions.

3.1.6.2 Configuration problems

Problem	Cause	Action

Opening the Preferences window fails and the following **Requirements** dialog is displayed: You will need a Java 1.4.1 or later compatible virtual machine installed on your system. The **Details** tab shows the version installed on your system. The **Info** tab shows the required version.	Your Java virtual machine version is earlier than the required version.	See the **Details** and **Info** tabs on the **Requirements** dialog for more information on the present and required versions of the Java virtual machine. Do the following To install the required virtual machine: 1 Go to the Source for Java Developers website at http://java.sun.com. 2 On the Source for Java Developers website, locate the J2SE SDK download link and click it. 3 Follow the download and installation instructions
Configuring global emulator settings fails.	The Preferences window is unable to write to the file.	If the file mentioned in the dialog does not exist, it means that the installation is corrupt and you need to reinstall the SDK.

3.1.6.3 Hints and tips

3.1.6.3.1 WM API

This S60 emulator supports Wireless Messaging API (WMA) specification 1.1 even though the javadoc version is 1.0 (no new classes since 1.0). A pdf version of the supported API specification is available at Wireless Messaging API Specification, Version 1.1 at http://jcp.org/aboutJava/communityprocess/final/jsr120/index2.html.

3.1.6.3.2 JAD/JAR checking

Real devices will perform MIDlet installation time checks to ensure consistency between the JAD file and the JAR file. Especially the manifest file in JAR needs to match with the JAD file as specified in the Mobile Information Device Profile (MIDP) Specification, Version 2.0 at http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html. The S60 emulator does not perform these checks. Care must be taken to ensure, that JAD and JAR file contents are in line with the specification. Otherwise a MIDlet may run in the emulator but fail to install into a real device.

3.1.6.3.3 MIDlet size limitations

The S60 emulator does not emulate the real memory size of a device. It should be kept in mind that there are limitations in real devices for maximum size of a MIDlet suite.

This emulator does not limit the amount of RMS data, that a MIDlet suite can store. However, real devices may have some device specific limitations for RMS data size. RMS data size also depends on the available memory and disk space on your PC. For example, if the amount of available disk space is 2 GB, the maximum RMS data size is 2 GB.

3.1.6.3.4 MIDP UI components

Command mapping

The implementation will add an **Exit** command into the **Options** menu if one is not provided by the application. For this reason it is sometimes possible to see two **Exit** commands in the **Options** menu. This happens if the application uses Command types incorrectly, that is, it has not used the proper Command.EXIT type for the **Exit** command.

Form

The **Copy text** and **Find in page** commands are added automatically into the displayable.

3.1.6.3.5 Button mappings

Basic navigation

Navigating the MIDP LCDUI high level UI elements is done with the navigation keys (up, down, left, right). Pressing OK or the key below Esc on the keyboard can be used as SELECT. LCDUI commands get mapped to the two soft buttons below the screen.

Game action mapping

Key 2 and 'Up' key	UP
Key 8 and 'Down' key	DOWN
Key 4 and 'Left' key	LEFT
Key 6 and 'Right' key	RIGHT
Key 5 and 'OK' key	FIRE
Key 7	GAME_A
Key 9	GAME_B
Key *	GAME_C
Key #	GAME_D

3.1.6.3.6 Common problems and solutions

IDE/debugger seems to freeze or become unresponsive during debugging

It often helps to switch to the emulator window and press a long press on the menu key followed by selecting the MIDlet process. This will switch the MIDlet into foreground and ensure it will get executed in the emulator. This symptom mostly happens during MIDlet startup and may be related to switching between `Displayables`.

Error message: Classpath is not set. Defaulting to "."

Check the command line parameters. This error message is sometimes given when there are unrecognized parameters on the command line.

Emulator does not start

Make sure that you have JAD and JAR files. In other words, the emulator cannot be started with pure MIDlet class.

The phone simulation used in this SDK is designed to show how applications may appear on J2ME™ enabled mobile phones. It simulates the look&feel and behavior of the S60 based phones as faithfully as possible, but there may be some differences between this simulator and the final products.

3.2 Using the SDK with an IDE

3.2.1 Supported IDEs

The following integrated development environments are supported by the S60 3rd Edition SDK for Symbian OS, Supporting Feature Pack 1, for MIDP:

- Carbide.j 1.5 (integrated with Eclipse)
- Eclipse 3.1.2 (or newer)
- NetBeans 5.0

Note: in order to be able to use NetBeans in conjunction with the S60 SDK, you also need to install the NetBeans Mobility Pack.

- IBM Websphere Studio Device Developer 5.7.1

3.2.2 Using the SDK with Eclipse/Carbide.j

3.2.2.1 Installing Eclipse

Prerequisites

Before installing Eclipse, make sure that you have Java™ 2 Platform (J2SE 1.4.2 (or newer)) installed on your PC.

J2SE 1.4.2 (or newer) can be downloaded from http://java.sun.com/downloads/index.html at http://java.sun.com/downloads/index.html.

Steps

1. Go to the Eclipse download website at http://www.eclipse.org/downloads/ at http://www.eclipse.org/downloads/.
2. Download the Eclipse SDK zip file to your PC.
3. Extract the contents of the downloaded zip file to a location of your choice on your PC.

Note:

The recommended way to install Eclipse is to extract the zip file contents to the root of your `C:` drive. in this case the Eclipse working directory will be `C:\eclipse`.

When extracting the contents of the zip file make sure that **Use folder names** is checked to preserve the directory structure of Eclipse.

Results

The installation is completed when you run `eclipse.exe` for the first time.

For more information about Eclipse, please refer to, for example, the Eclipse FAQ website at http://wiki.eclipse.org/index.php/Eclipse_FAQs.

3.2.2.2 Installing Carbide.j

Context

Since the Carbide.j as a standalone application does not include tools for editing, compiling or debugging the MIDlet and personal profile classes, it is here recommended that you install Carbide.j integrated with the Eclipse IDE, which provides tools for editing, compiling and debugging MIDlets. Thus, these installation instruction presume that you already have Eclipse installed on your PC. (For more information about Eclipse, please refer to http://www.eclipse.org at http://www.eclipse.org/, where you can also download the latest Eclipse SDK.)

Steps

1 Launch the Carbide.j installer.

. The Introduction pane of the installer is displayed:

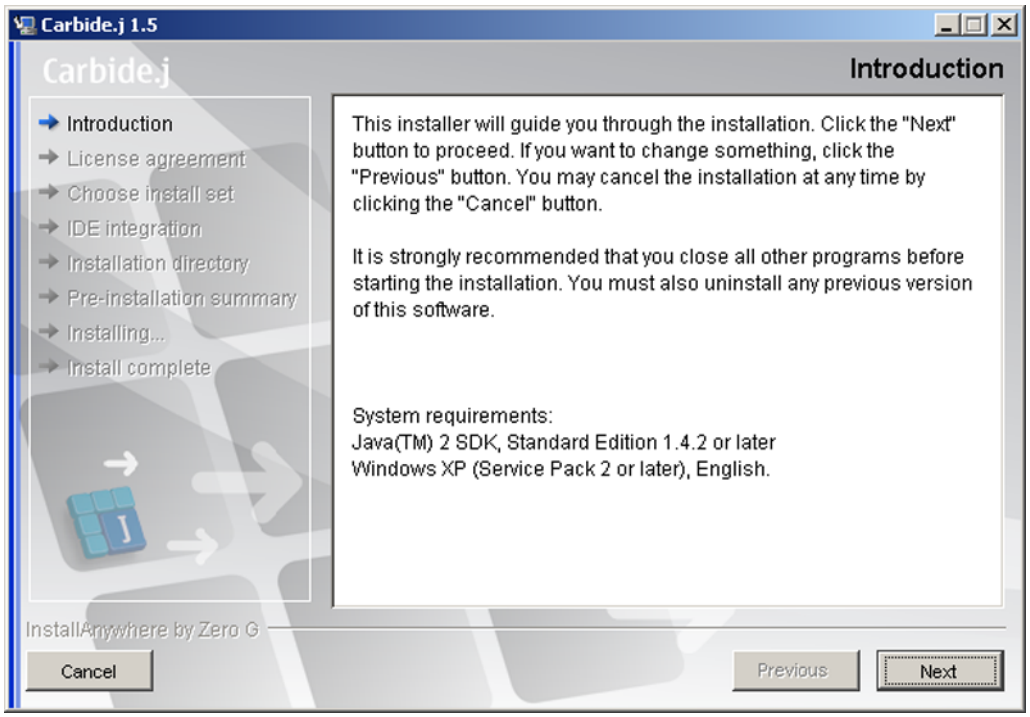


Figure 63: Introduction pane

2 In the Introduction pane, click **Next**.

. The Carbide.j license agreement pane is displayed:

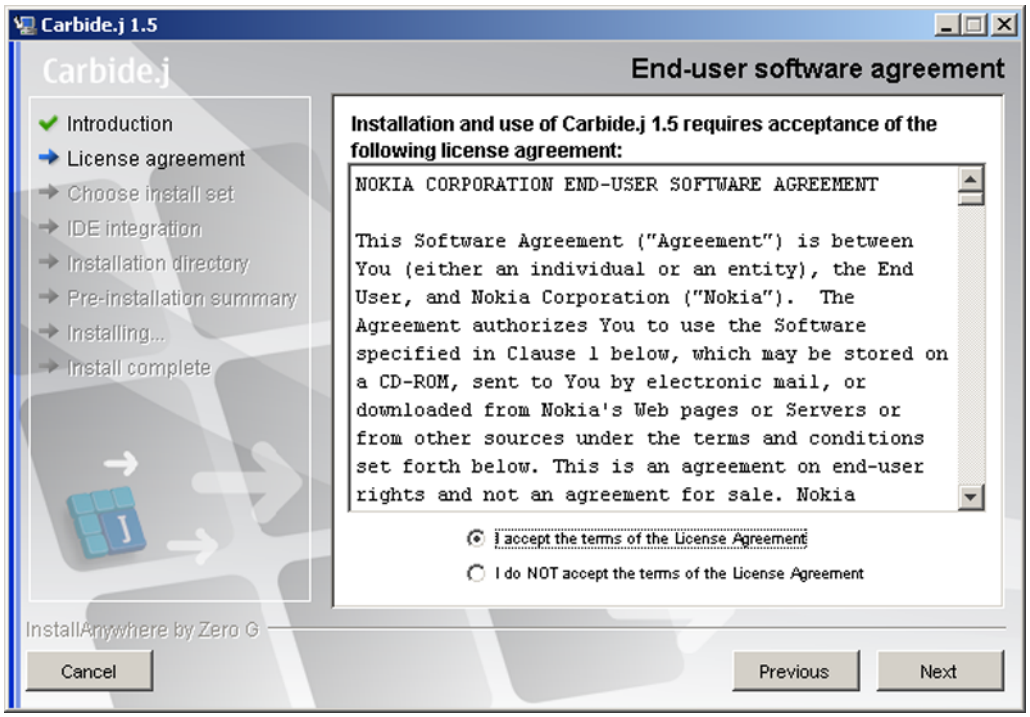


Figure 64: Carbide.j license agreement

3. In the License agreement pane, select "I accept the terms of the License Agreement" and click **Next**.

   The **Choose install set** pane is displayed:

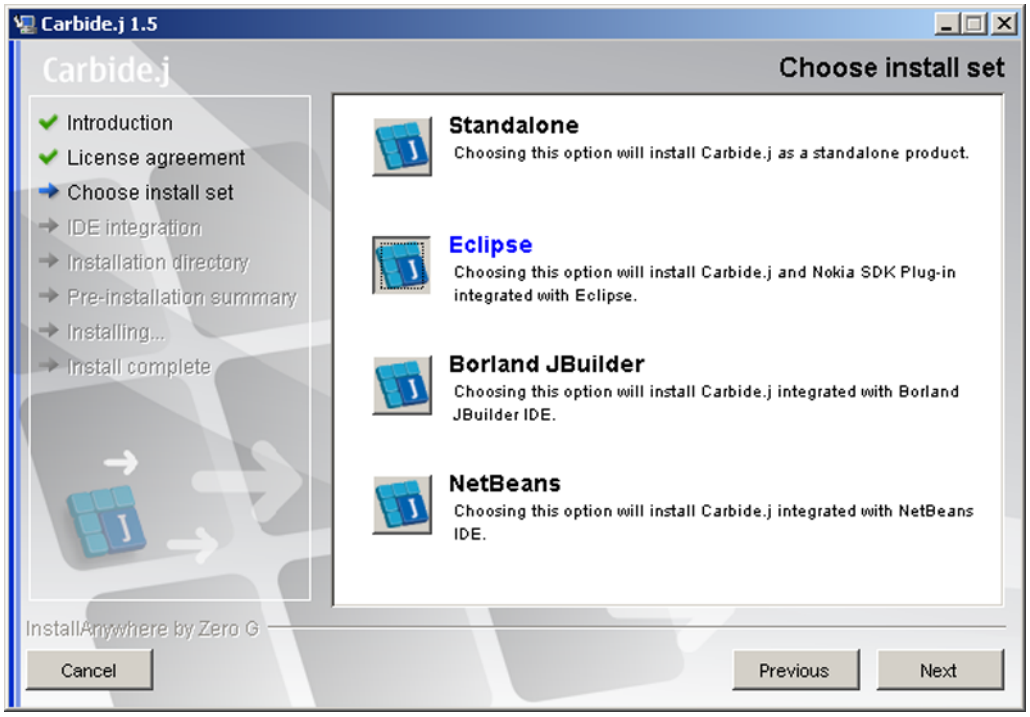


Figure 65: Choose install set

4. In the **Choose install set** pane, select "Eclipse".

   The **IDE integration** pane is displayed:

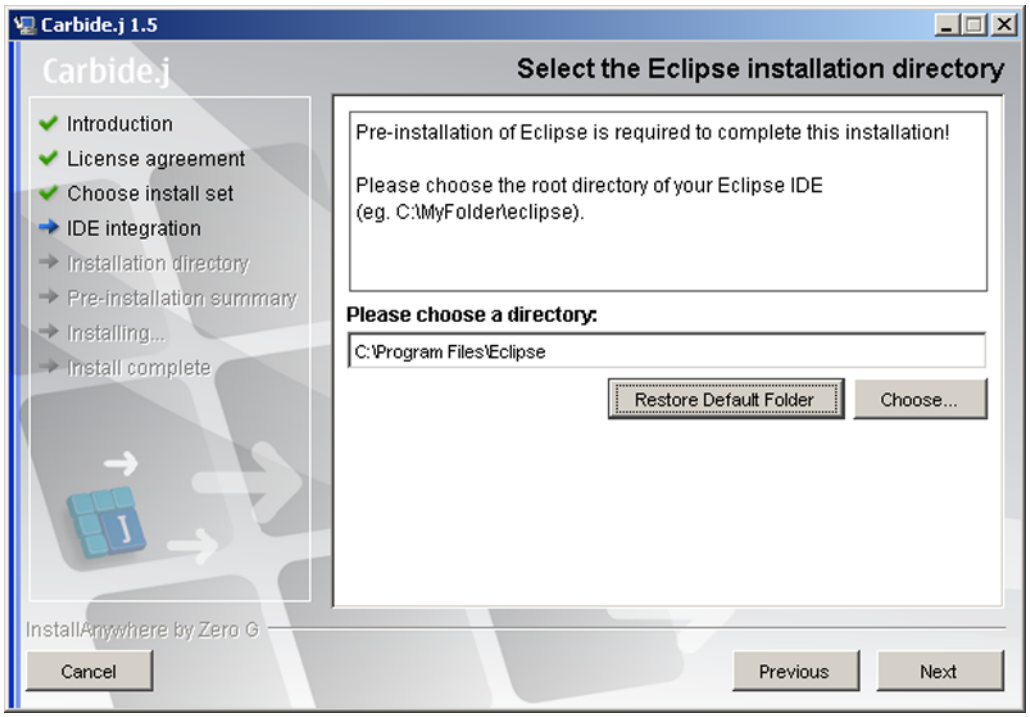


Figure 66: IDE integration

5. In the IDE integration pane, click **Choose...**, browse to your Eclipse installation directory and click **Next**.

   The **Installation directory** pane is displayed:

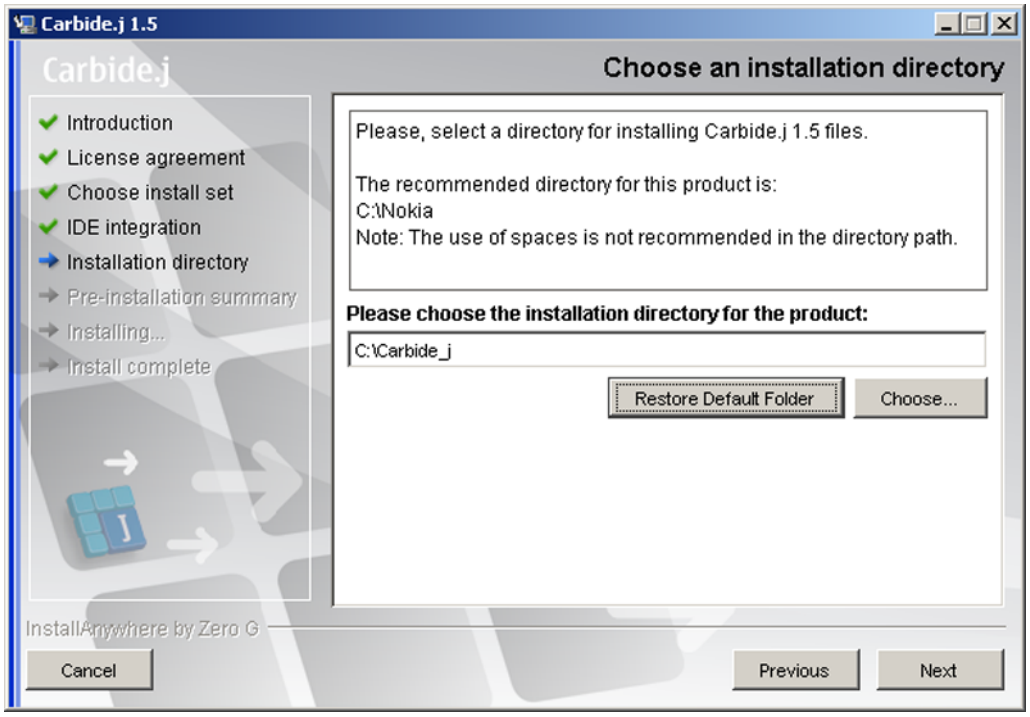


Figure 67: Installation directory

6. In the **Installation directory** pane, either accept the default directory suggested or browse to a directory of your own choice by cliking **Choose....**. Then click **Next**.

   The **Pre-installation summary** pane is displayed:

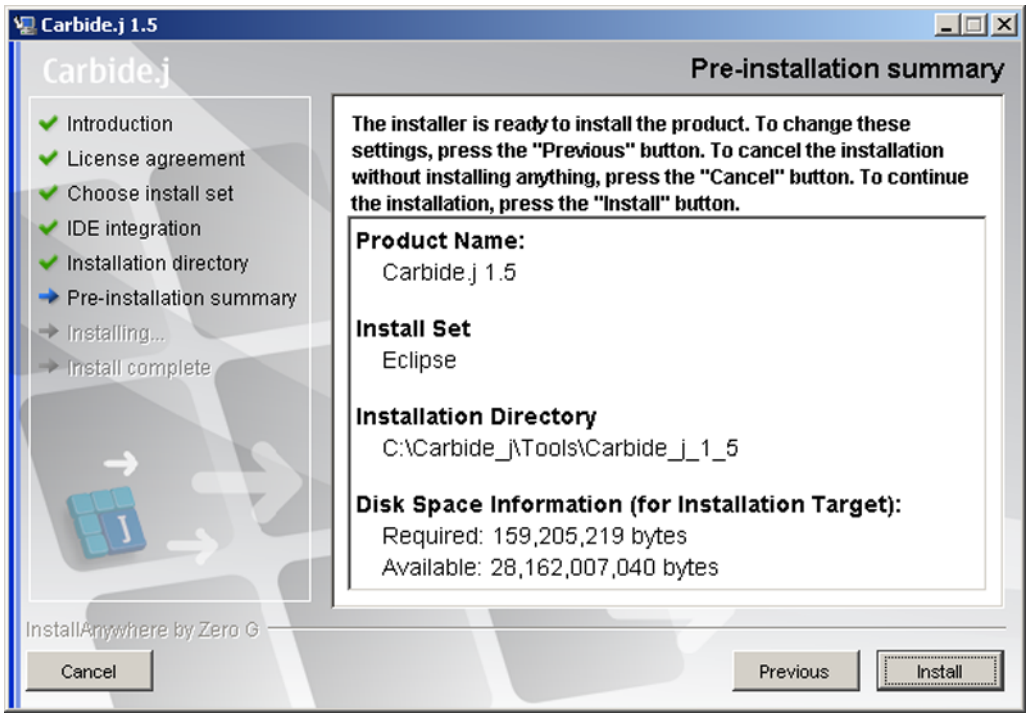


Figure 68: Pre-installation summary

7 In the **Pre-installation summary** pane, click **Install**.

. Carbide.j installation starts:

Figure 69: Installing

Once the installation is ready, the Install complete pane is displayed:

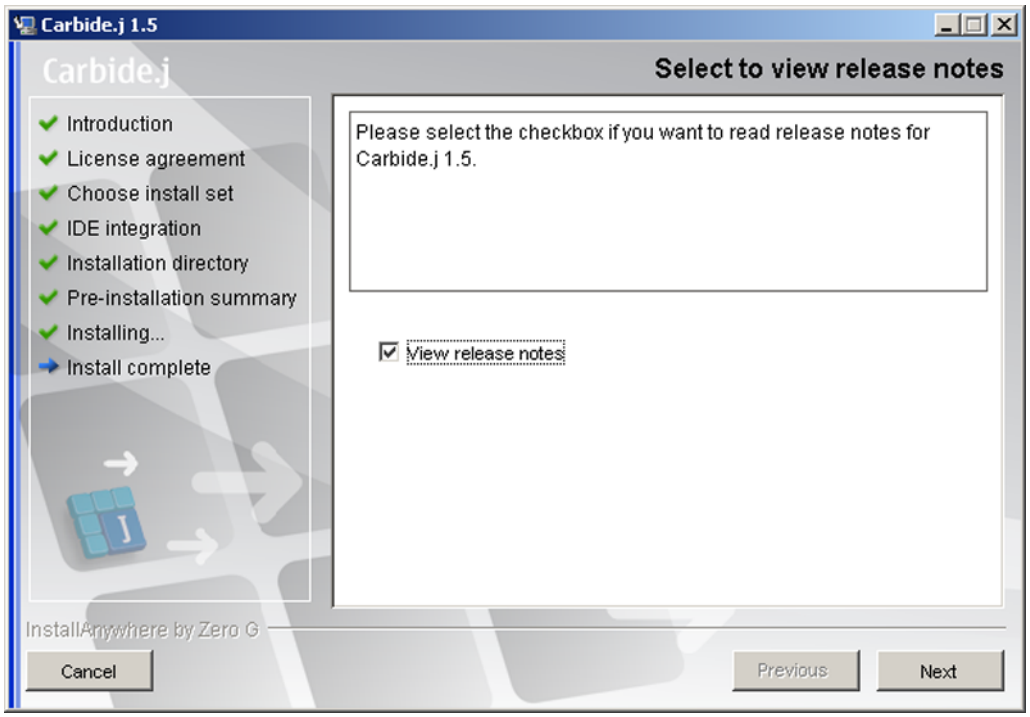


Figure 70: Install complete

	Note: You need to restart your PC after installation in order to get all Carbide.j features running.

Results

Carbide.j has now been installed, integrated with the Eclipse IDE. You can access Carbide.j by opening Eclipse and selecting **Tools > Carbide.j** from the Eclipse menu bar. All Carbide.j functionality is provided through Eclipse.

	Note: While integrated with Eclipse, carbide.j can also be opened and run as a standalone application from the windows **Start** menu by selecting **Start > All Programs > Carbide > Carbide.j 1.5 > Run as Standalone** . The Carbide.j User's Guide is also accessible through the Windows **Start** menu.

3.2.2.3 Setting Java debug preferences for Eclipse

Context

When debugging MIDlets on the S60 SDK emulator, it is necessary to turn off some debugger preferences in Eclipse.

Steps

1. Select **Window > Preferences** from the **Eclipse** menu.
2. Select **Java > Debug** from the **Preferences** tree.
3. Clear checkboxes **Suspend execution on uncaught exceptions**, **Suspend execution on compilation errors** and **Suspend for breakpoints during evaluation**.

4. Increase timeout values to prevent debugger timing out before the debugged MIDlet has started.

   We recommended that you increase the debugger timeout to 50,000 when debugging with S60 MIDP SDK.

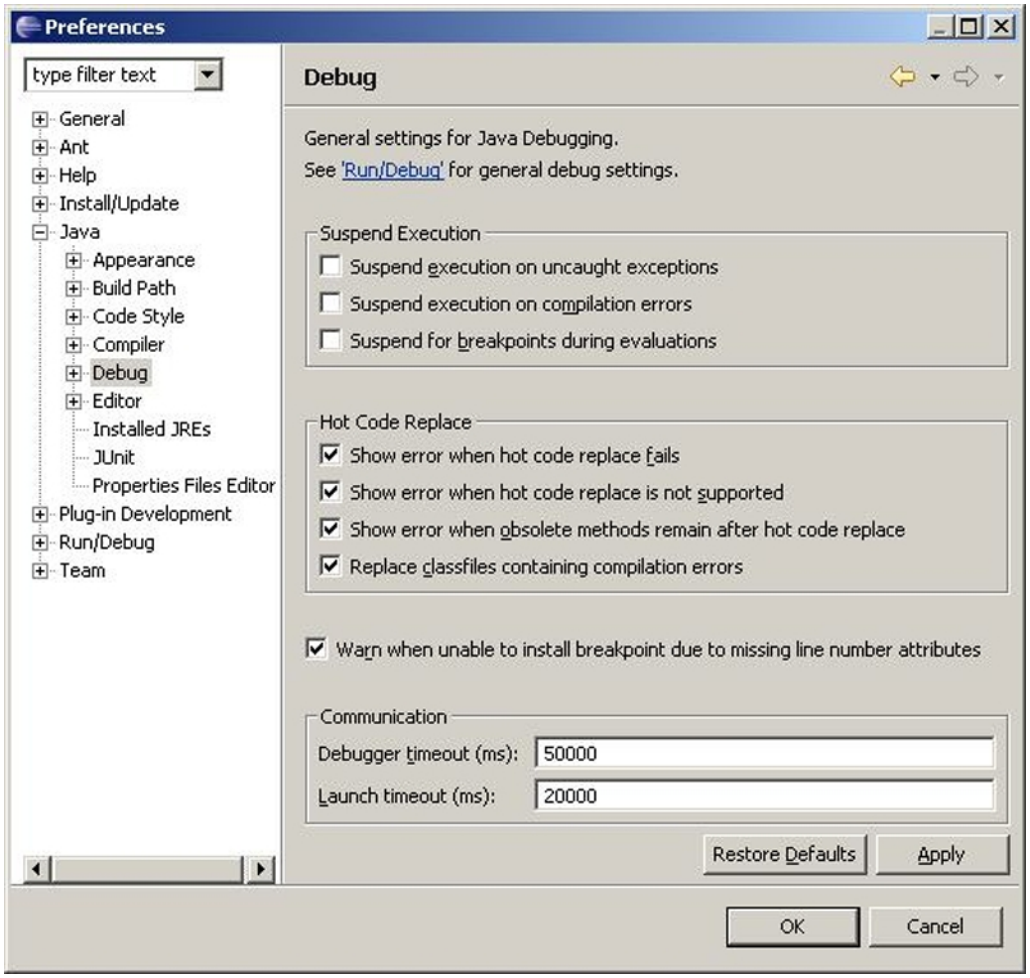


Figure 71: Java Debug Settings for Eclipse

3.2.2.4 Configuring the S60 SDK emulator for Carbide.j

Context

Carbide.j comes with Nokia Prototype SDK 4.0 for Java™ ME emulators. If you want to use the S60 SDK emulator, you have to configure this for Carbide.j separately.

Steps

1. Open Eclipse.
2. Select **Tools > Carbide.j > Configure Emulators...** from the Eclipse menu bar.
3. The **Configure Emulators** pane is displayed:

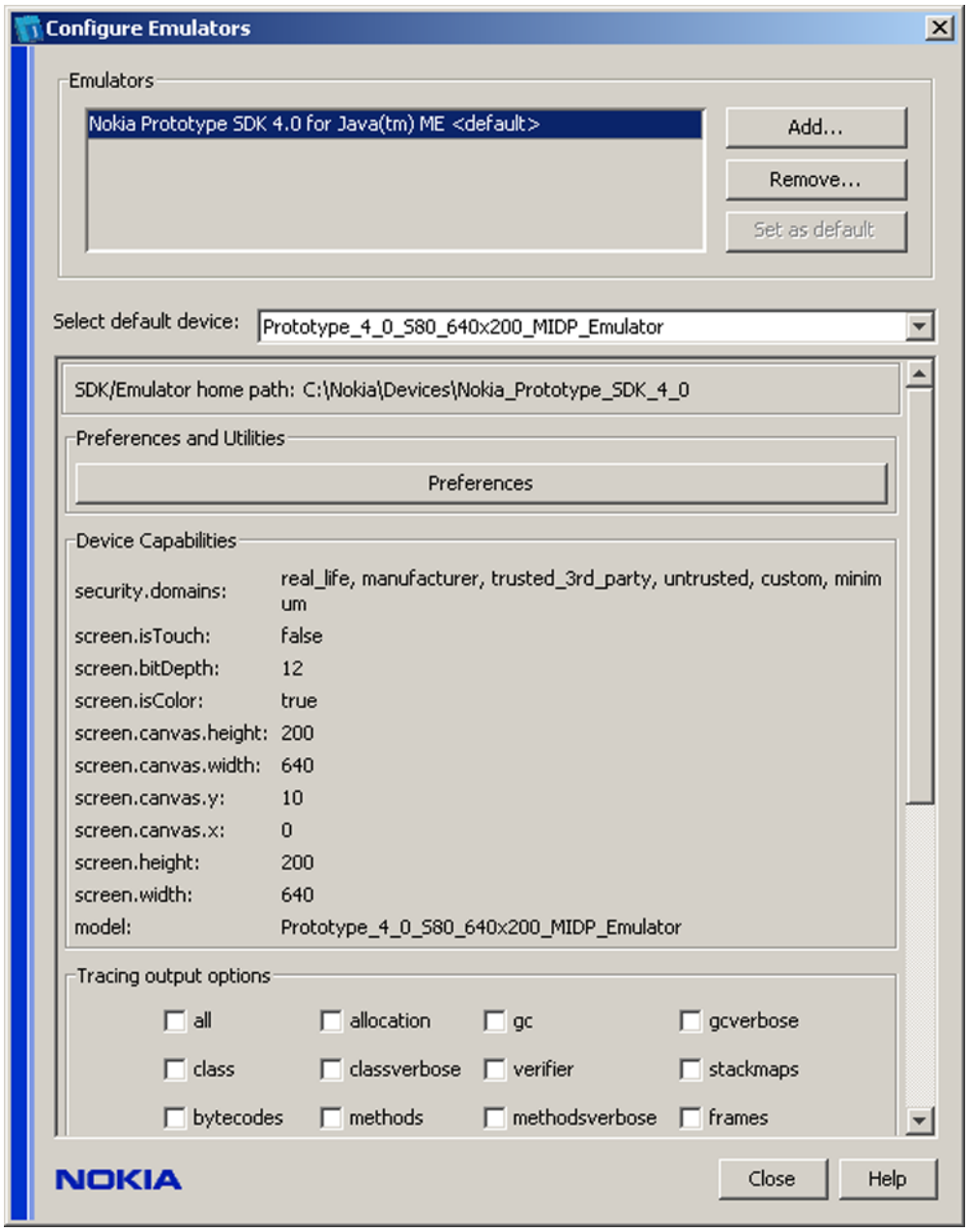


Figure 72: Configure Emulators pane

3 In the **Configure Emulators** pane, click **Add**.

.

4 In the **Open** dialog, browse to the SDK installation directory and click **Open**.

. The S60 MIDP SDK emulator has now been added to the list of supported emulators:

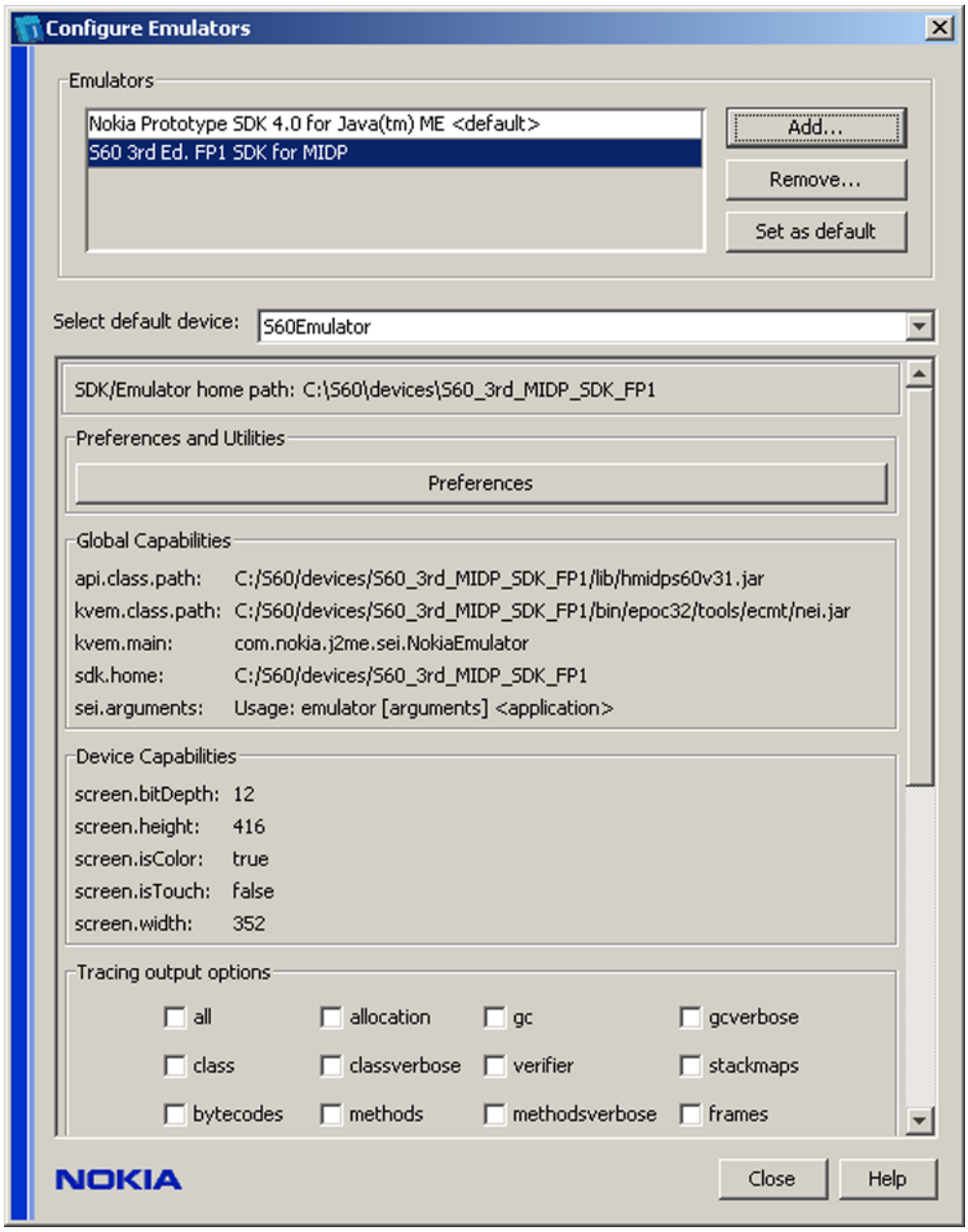


Figure 73: Supported emulators

5. Set the S60 MIDP SDK emulator as the default emulator by clicking **Set as default**.

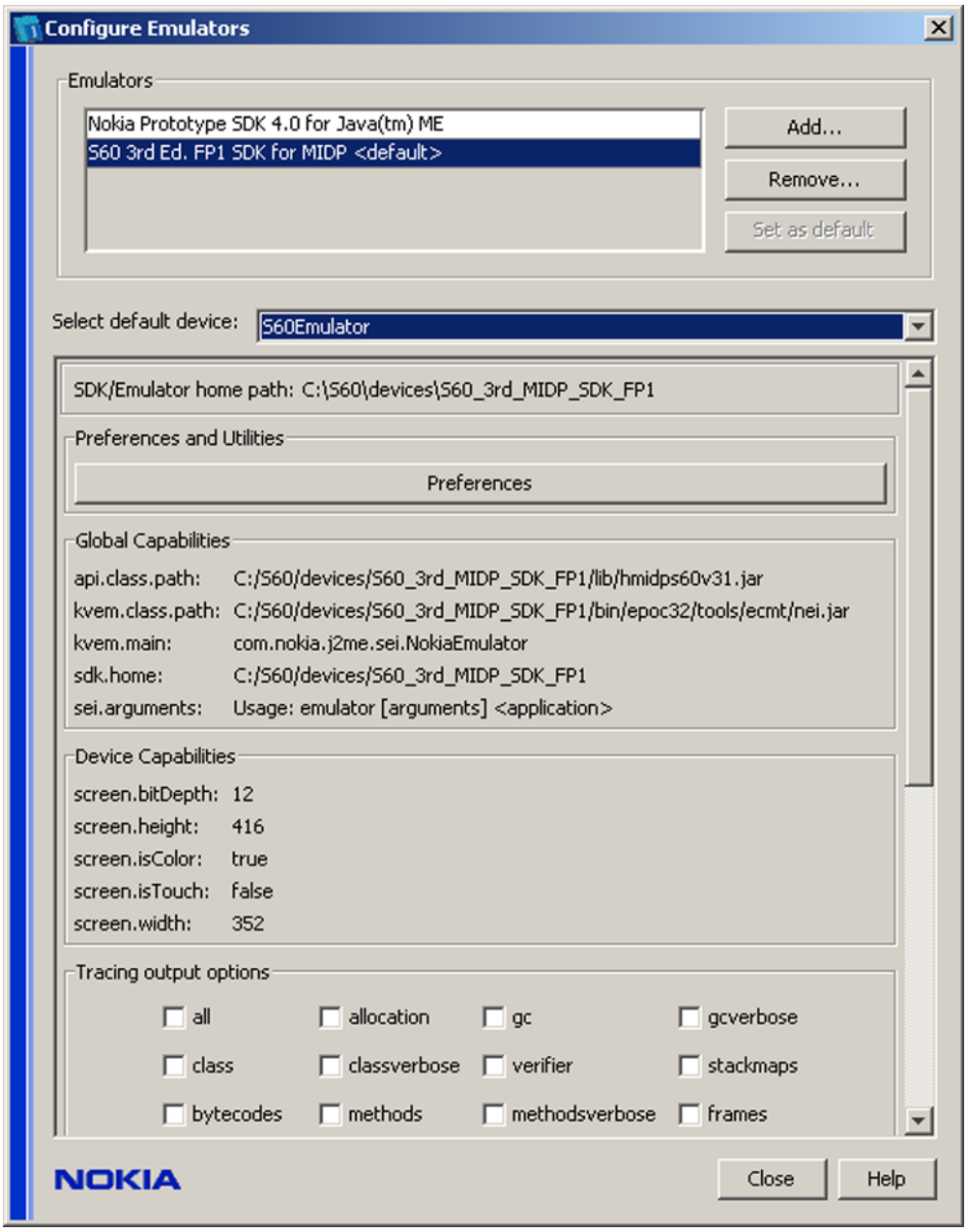


Figure 74: S60 MIDP SDK as default emulator

6 . You can set emulator preferences in the **Preferences and Utilities** section of the **Configure Emulators** pane.

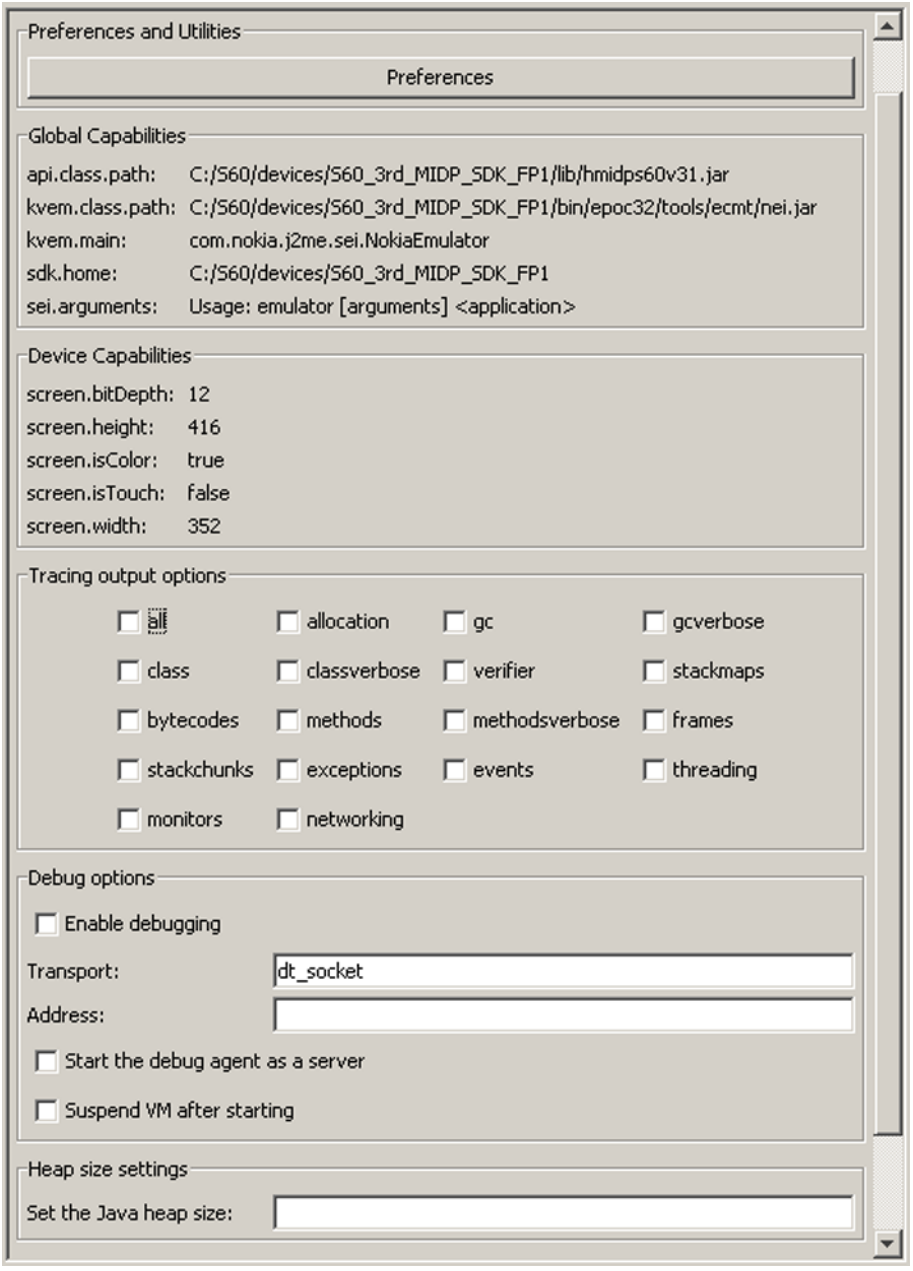


Figure 75: Emulator preferences and utilities

Results

The S60 SDK emulator has now been set as the default emulator for Carbide.j. In other words, whenever you run a MIDlet from Carbide,j, it will be opened in the S60SDK emulator.

3.2.2.5 Running an S60 SDK example MIDlet from Carbide.j

Context

The S60 SDK comes with a number of example MIDlets, which can be used in application development. The MIDlets can also be run on the S60 emulator from Carbide.j.

Note: In order to follow the instructions here, the following prerequisites have to be in place

- S60 3rd Edition SDK for Symbian OS, Supporting Feature Pack 1, for MIDP (S60 SDK) has been installed
- Carbide.j has been installed and integrated with Eclipse (see Installing Carbide.j)
- The S60 SDK emulator has been set as the default emulator for Carbide.j (see Configuring SDK emulator for Carbide.j).

Steps

1. Open Eclipse.
2. Select **Tools > Carbide.j > Start Emulators...** from the Eclipse menu bar.

   The **Start Emulators** pane is displayed:

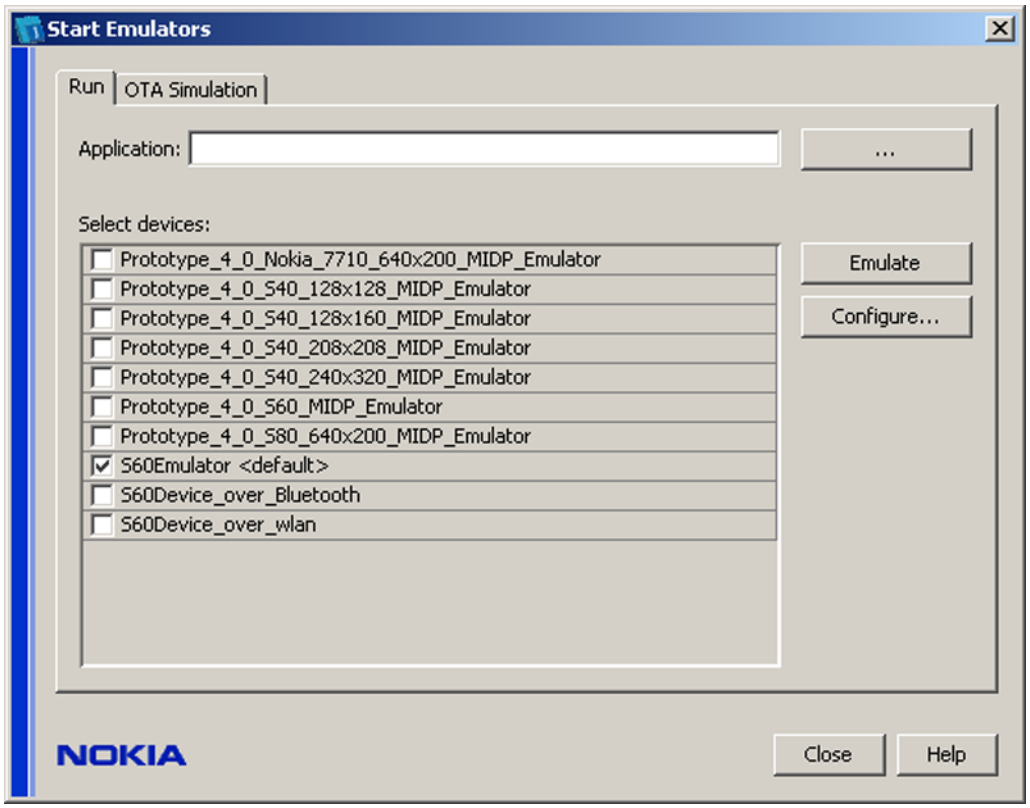


Figure 76: Start Emulators pane

Note: Make sure that the S60 SDK emulator is the default emulator (as in the figure above).

3. In the **Start Emulators pane** pane, click the **...** button.
4. In the **Open** dialog, browse to the `\S60examples` folder in the SDK installation directory and click **Open**.
5. Go to the `\bin` folder of the example application that you want to run and click the `.JAD` file located there.

   For example: `<S60_SDK_installation_directory>\S60_3rd_MIDP_SDK_FP1\S60examples\helloworldplus\bin\helloworldplus.jad`.

6 Click **Open**.

. The `helloworldplus.jad` file is opened into the **Start Emulators** dialog, as displayed in the **Application:** field

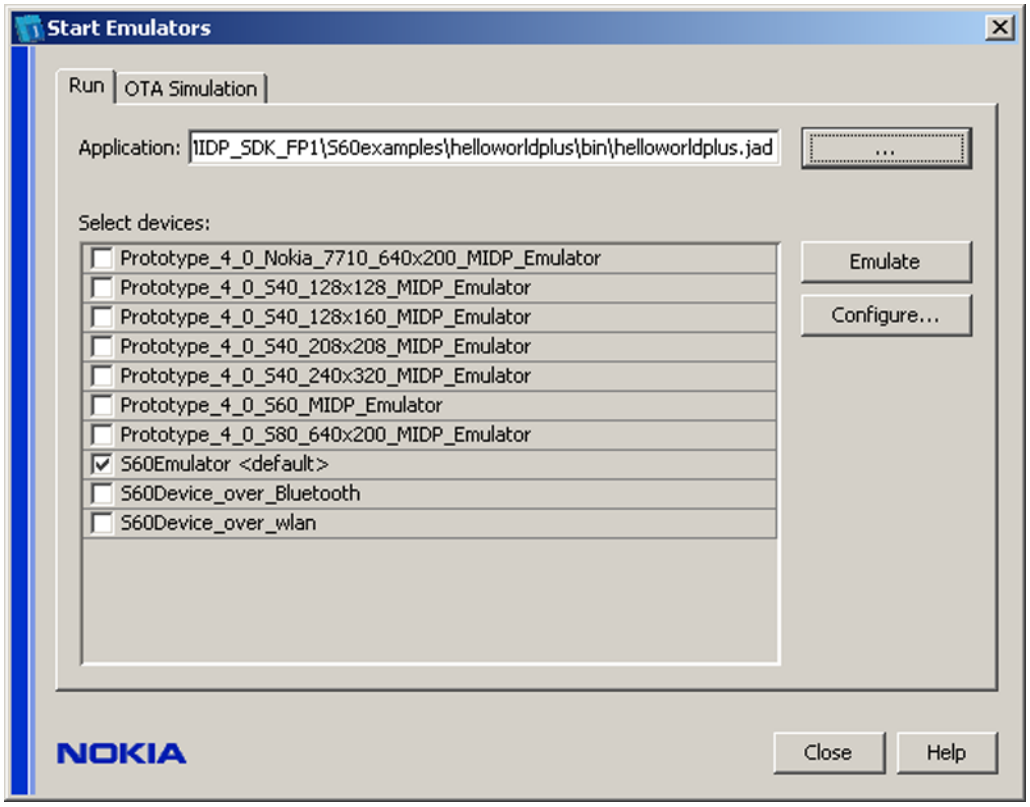


Figure 77: JAD file in Application: field

7 Click the **Configure…** button if you want to define tracing output, debug and heap
. size options before running the application in the emulator (see Emulator preferences and utilities on page 0 for more information).

8 Click **Emulate** in the **Start Emulators** pane.

.

Results

The example application in question is launched in the S60 SDK emulator.

Note: It may take some time for the emulator to start.

To locate the HelloWorldPlus application, do the following:

1 Click the Applications button to open the emulator's application grid (see Graphical user interface).

2 In the emulator's application grid, use the five-way navigation key to open the **Installed** folder.

3 In the **Installed** folder, open the HelloWorldPlus application.

3.2.2.6 Creating a new project with Carbide.j

The stand-alone version of Carbide.j does not maintain any project-related information as such. Thus the instructions provided in this Help here describe how to use Carbide.j integrated into the Eclipse IDE (as instructed in Installing Carbide.j). For information on how to create a new project with Carbide.j, please refer to Creating a new MIDP project in Eclipse.

For information on using Carbide.j as a standalone application, please refer to the Carbide.j *Users Guide*, which you can access through the Windows Start menu: **Start > All Programs > Carbide > Carbide.j 1.5 > Users Guide**.

3.2.3 Using the SDK with Netbeans

3.2.3.1 Installing and configuring NetBeans

Prerequisites

In order to integrate NetBeans IDE with the S60 SDK you also need to install NetBeans Mobility Pack.

Steps

1. Install NetBeans (please refer to http://www.netbeans.org/downloads/index.html at http://www.netbeans.org/downloads/index.html).
2. Install the NetBeans Mobility Pack (please refer to http://www.netbeans.org/products/mobility/ at http://www.netbeans.org/products/mobility/).
3. Start NetBeans and select **Tools > Java Platform Manager**.

   The **Java Platform Manager** dialog opens:

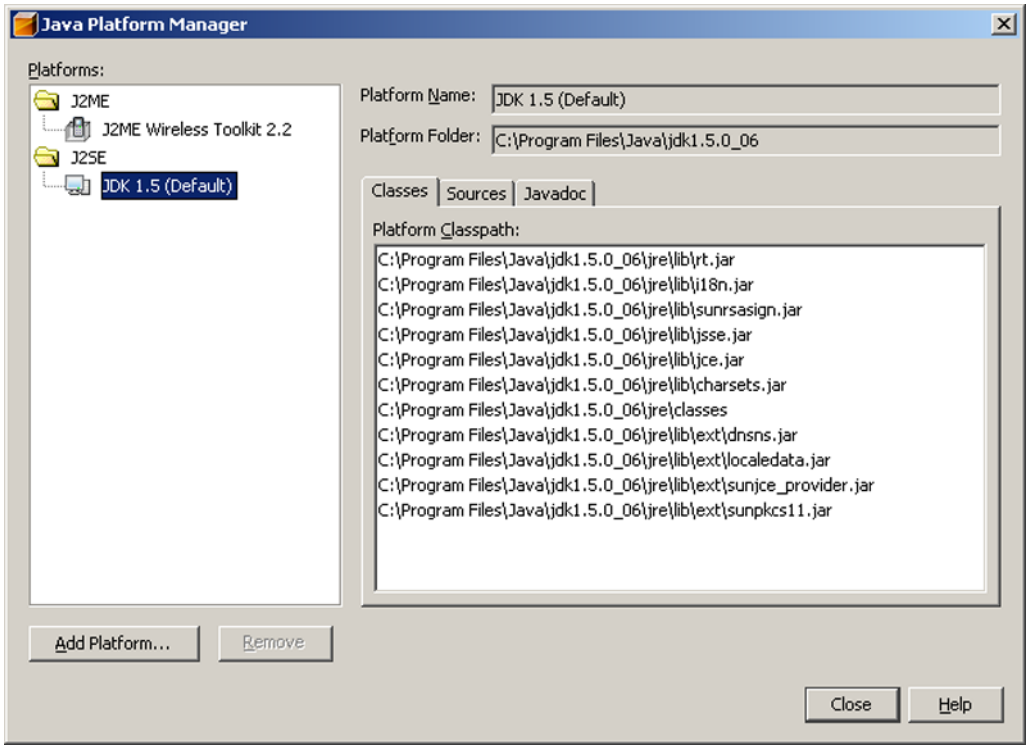


Figure 78: Java Platform Manager

4. In the Java Platform Manager, click **Add Platform...**.

   The **Select platform type** view opens:

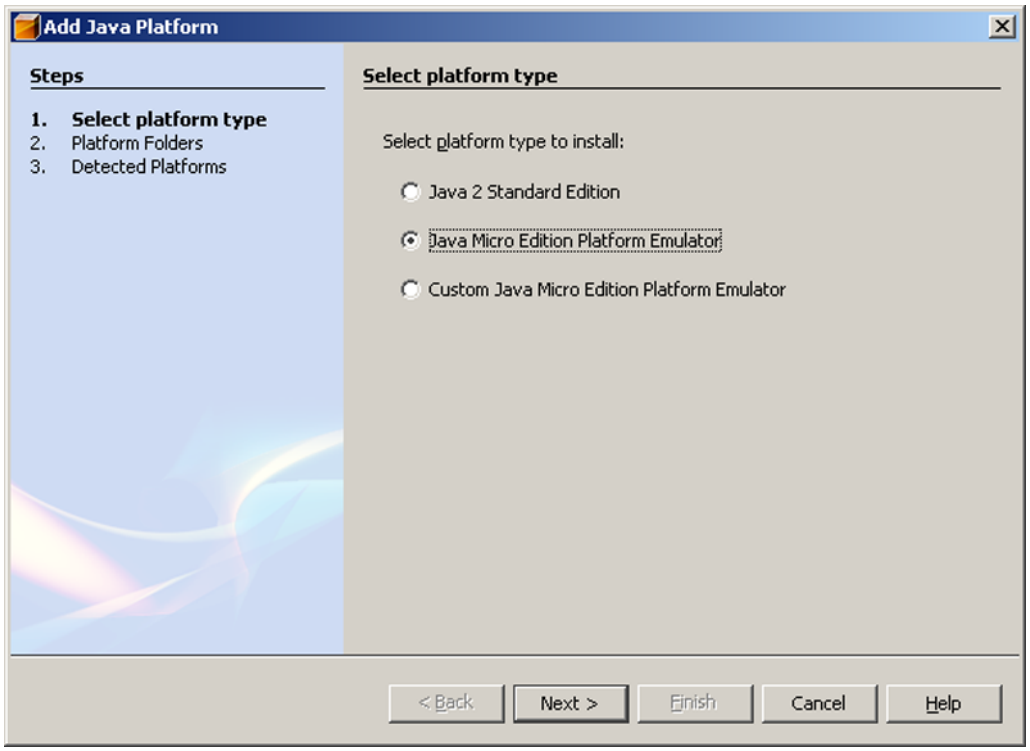


Figure 79: Select platform type

5. In the **Select platform type** section, select **Java Micro Edition Platform Emulator** and click **Next**.

   The **Platform Folders** view opens:

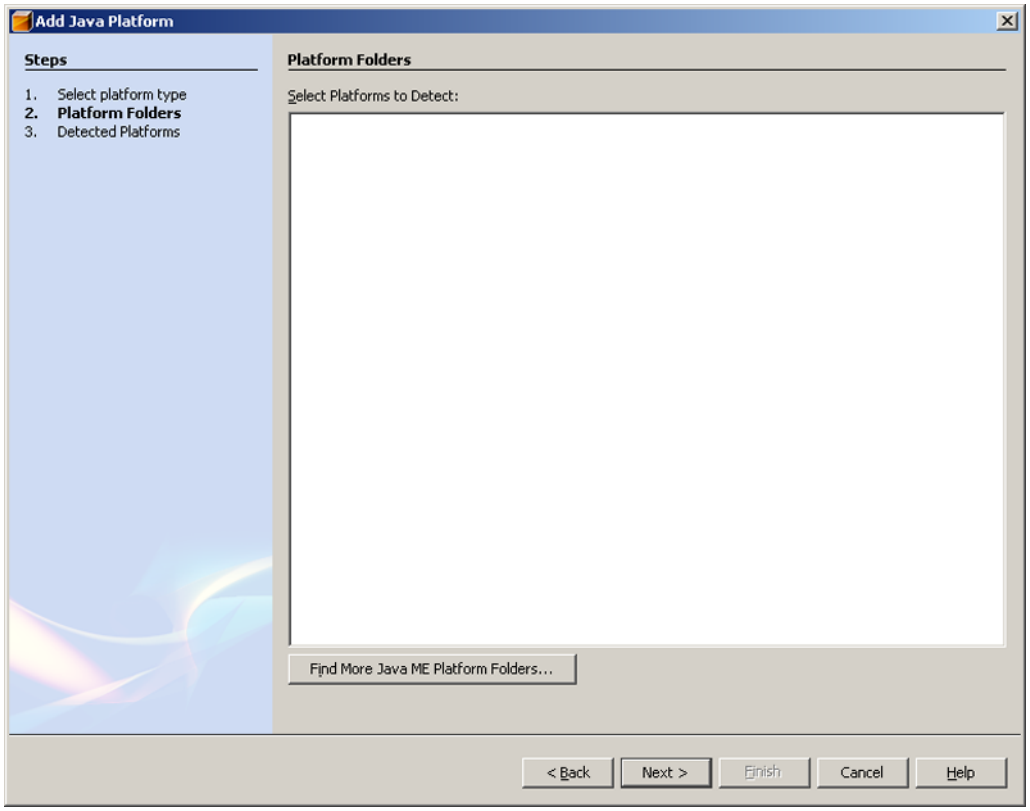


Figure 80: Platform folders

6. In the **Platform Folders** view, click **Find More Java ME Platform Folders…**.

7. In the **Choose directory to search for platforms** dialog that appears, browse to the SDK installation directory (e.g. `<S60_SDK_installation_directory>\S60_3rd_MIDP_SDK_FP1`) and click **Search**.

   NetBeans locates the S60 SDK:

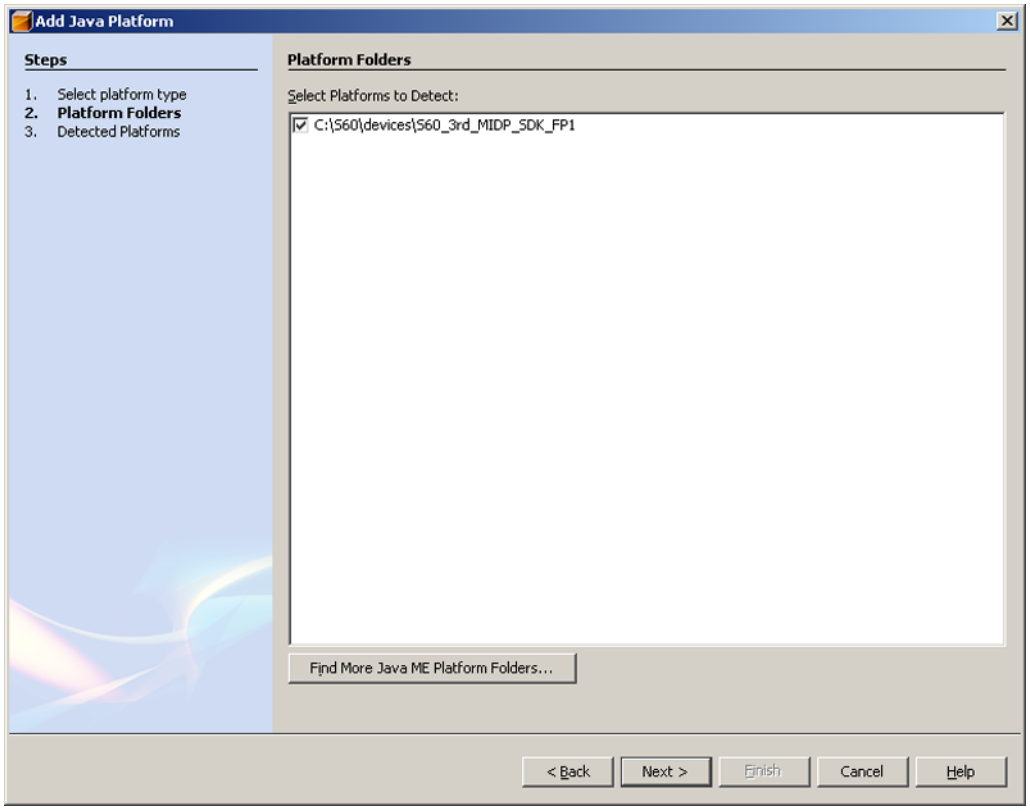


Figure 81: S60 SDK in discovered platform folders

8. Make sure the S60 SDK installation is checked in the **Platform Folders** list and click **Next**>.

   NetBeans detects the S60 SDK platform, as displayed in the list of detected platforms, which appears:

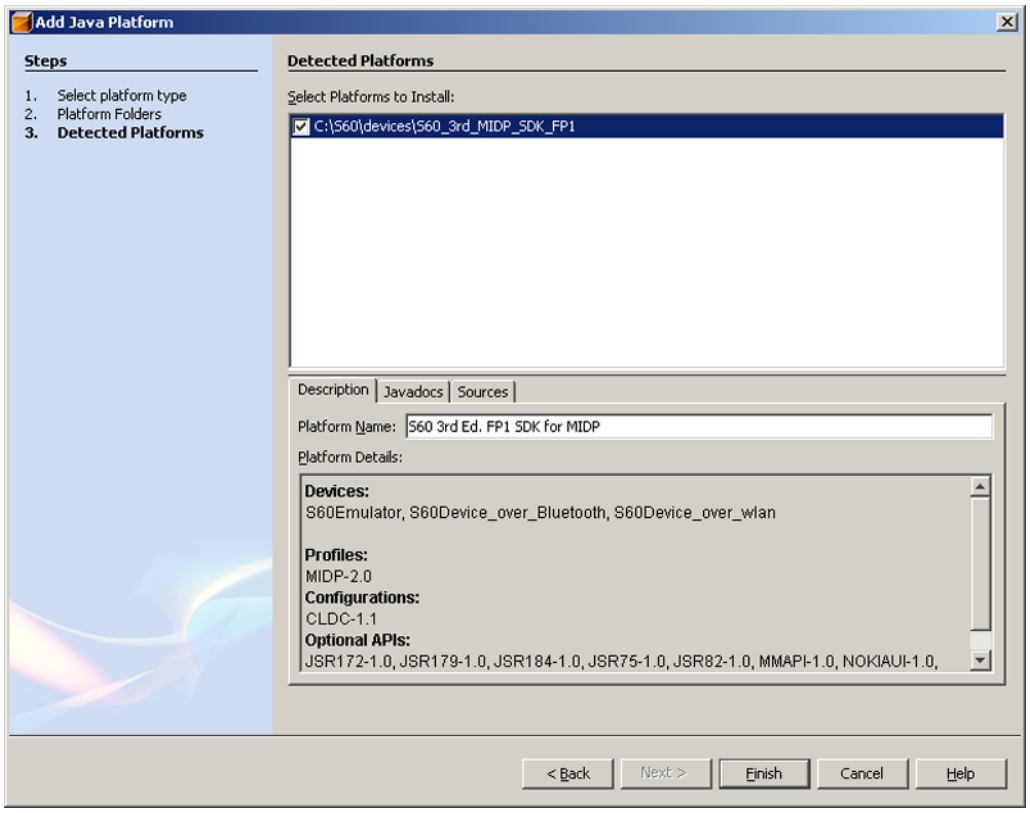


Figure 82: Detected platforms

9. Click **Finish**.

   The S60 SDK is added to the detected platforms in NetBeans:

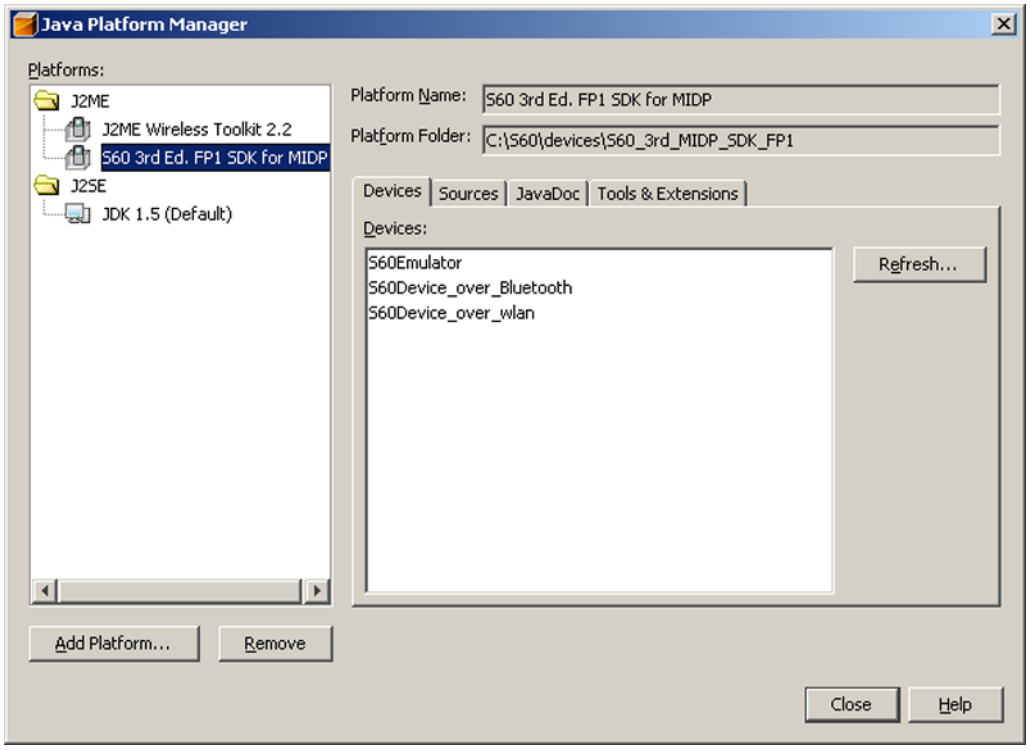


Figure 83: S60 SDK added to Java Platforms

10. Click **Close**.

Results

The S60 SDK has now been added to the NetBeans IDE.

3.2.3.2 Running an S60 SDK example MIDlet from NetBeans

Context

The S60 SDK comes with a number of example MIDlets, which can be used in application development. The MIDlets can also be run on the S60 emulator from NetBeans.

	Note: In order to follow the instructions here, the following prerequisites have to be in place • S60 3rd Edition SDK for Symbian OS, Supporting Feature Pack 1, for Java (S60 SDK) has been installed • NetBeans and NetBeans Mobility Pack have been installed, and the S60 SDK has been configured for NetBeans (see Installing and configuring NetBeans)

Steps

1. Open the NetBeans IDE.
2. Select **File > Open project…** from the menu bar.

   The **Open Project** view is displayed:

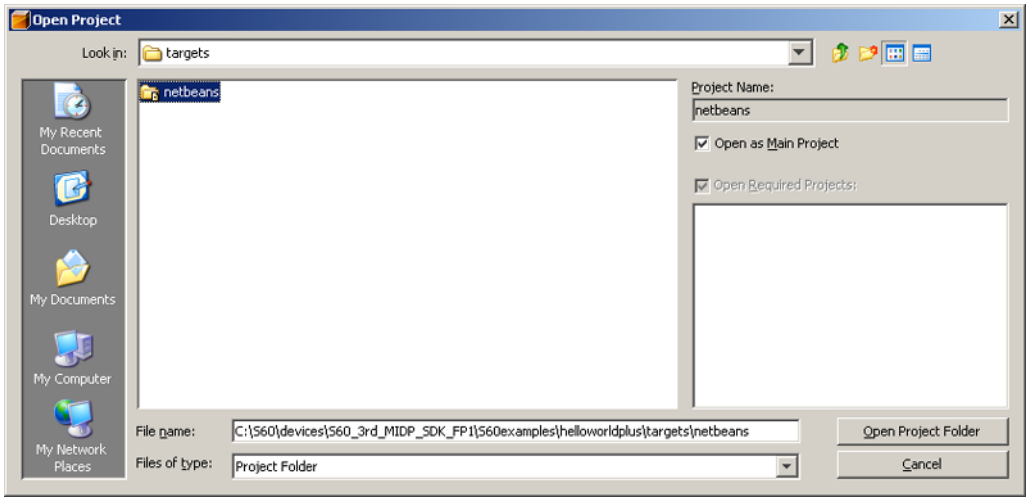


Figure 84: Open project

3. Browse to the NetBeans project file located in the in the `\targets\netbeans` folder of the S60 example application that you want to run.

   For example, `<S60_SDK_installation_directory>\S60examples\helloworldplus\targets\netbeans`

4. Click **Open Project Folder**.

   The NetBeans project is opened in the NetBeans IDE.

5. Click the **netbeans** project icon in the project view on the left and select **Properties** from the pop-up menu.

   The **Project configuration** view is displayed:

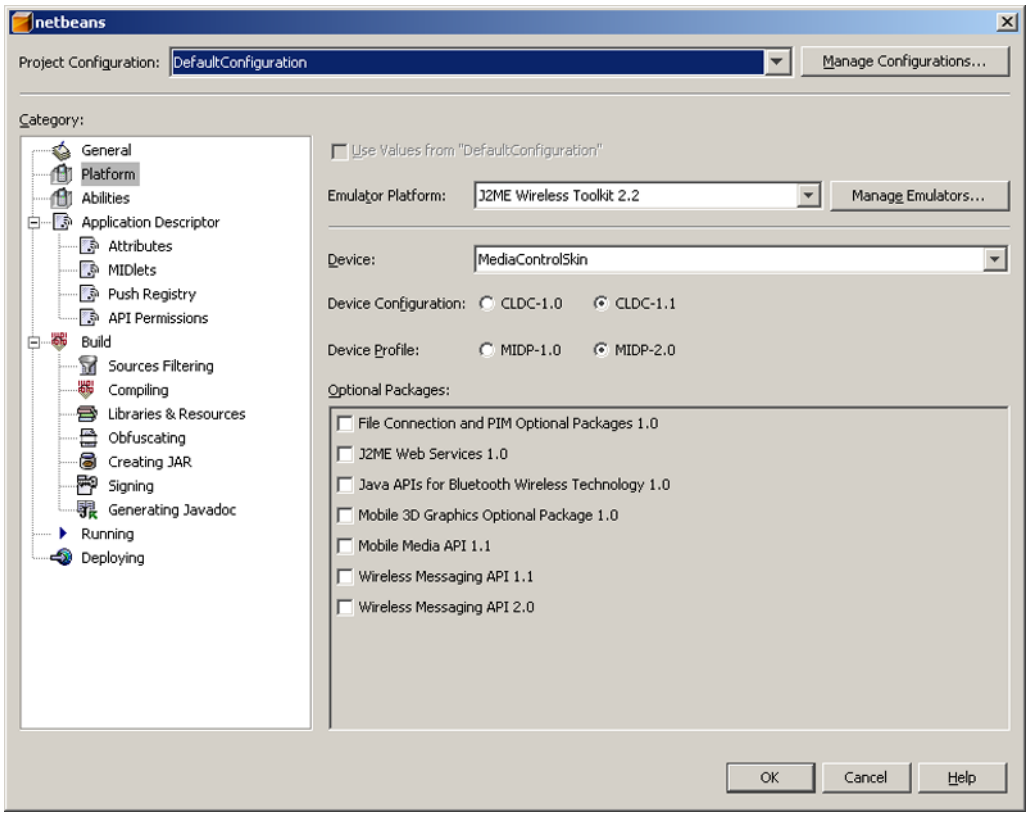


Figure 85: Project configuration view

Here you need to set the S60 SDK emulator as the default project configuration.

6. Click **Manage Configurations…**.
7. In the **Project Configuration Manager** view that is displayed, click **Add…**.
8. In the **Add Project Confugration** view that is displayed, select "S60_emulator_template" from the **Use Configuration Template** drop-down menu and click **OK**.

   The S60Emulator configuration is added to the list of project configurations.
9. In the **Project Configuration** view, click **Close**.
10. With the S60Emulator in the **Project Configuration** field of the **Project configuration** view, click **OK**.
11. Select **Build > Build Main Project** from the menu bar.

    The build output is displayed in the **Output** field at the bottom of the IDE.
12. Select **Run > Run Main Project** from the menu bar.

Results

The S60 SDK emulator is launched.

Note: It may take some time for the emulator to start.

To locate the S60 example application, do the following:

1 Click the Applications button to open the emulator's application grid (see Graphical user interface).

2 In the emulator's application grid, use the five-way navigation key to open the **Installed** folder.

3 In the **Installed** folder, open the application.

3.2.3.3 Creating a new project with NetBeans

Steps

1 Open the NetBeans IDE.

.

2 Select **File > New Project**.

. The **New Project** dialog opens:

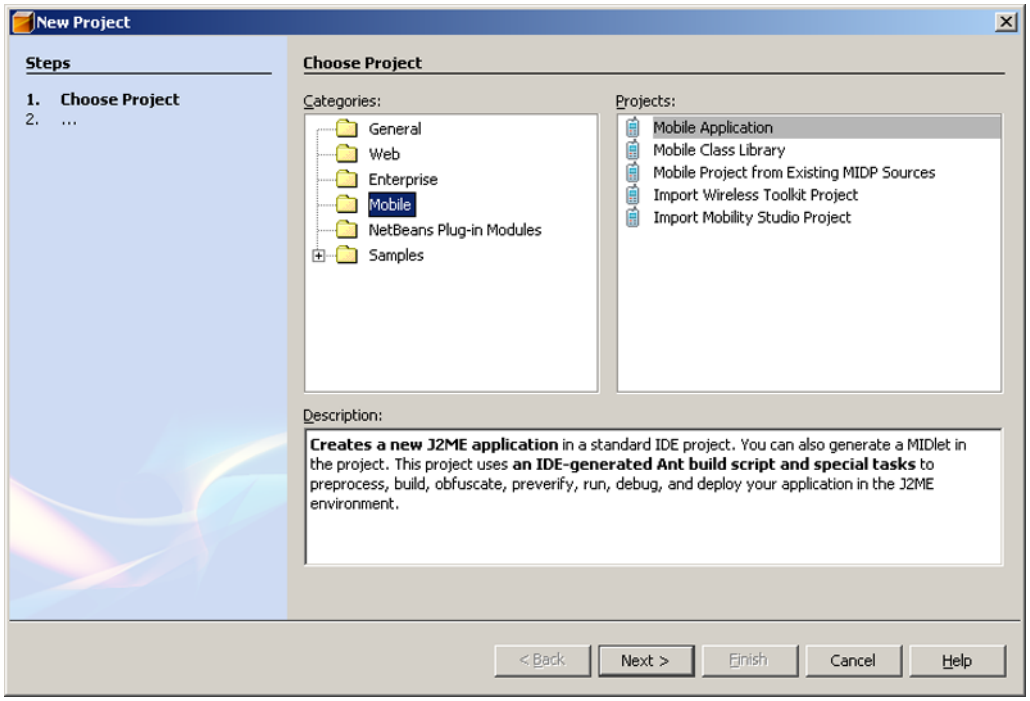


Figure 86: New Project window

3 In the **Categories:** pane, select **Mobile**.

.

4 In the **Projects:** pane, select **Mobile Application** and click **Next**.

. The **Name and Location** view opens:

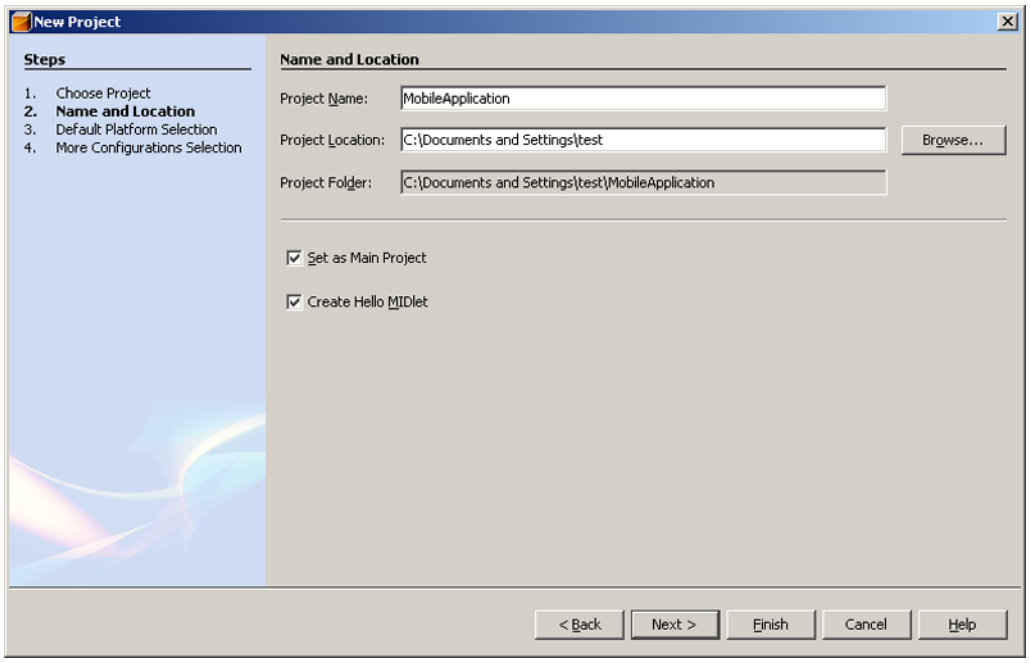


Figure 87: Project name and location

5 Define the new project name and location and click **Next**.

. The **Default Platform Selection** view opens.

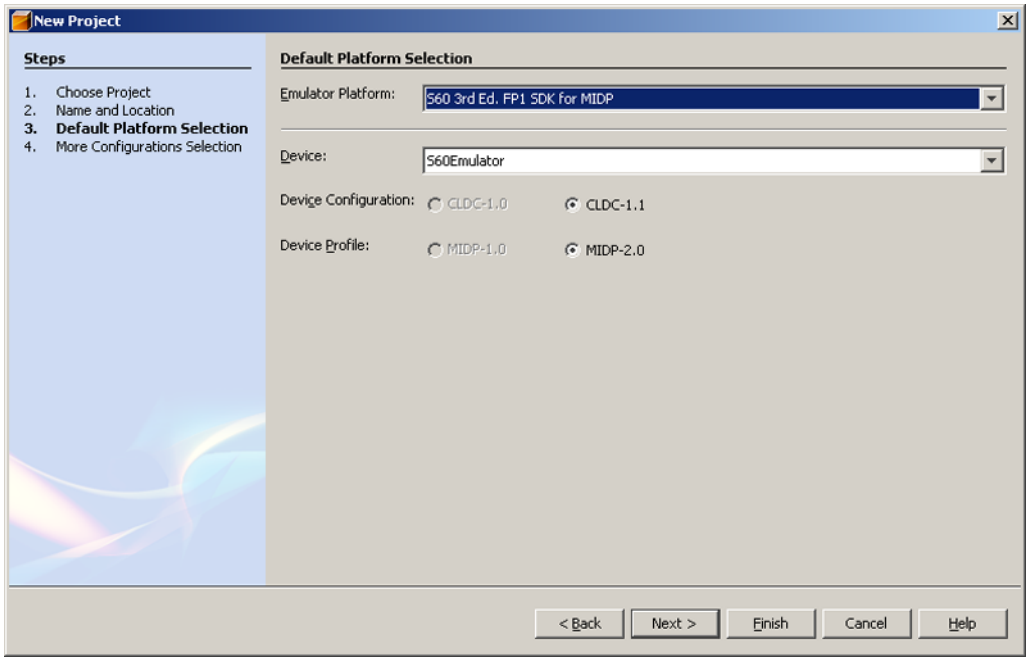


Figure 88: Platform Selection dialog

6 Select "S60 3rd Ed. FP1 SDK for MIDP" in the **Emulator Platform:** drop down menu.

.

7 Select "S60Emulator" in the **Device:** drop down menu.

.

8 Clcik **Next**.

. The **More Configurations Selection** view opens:

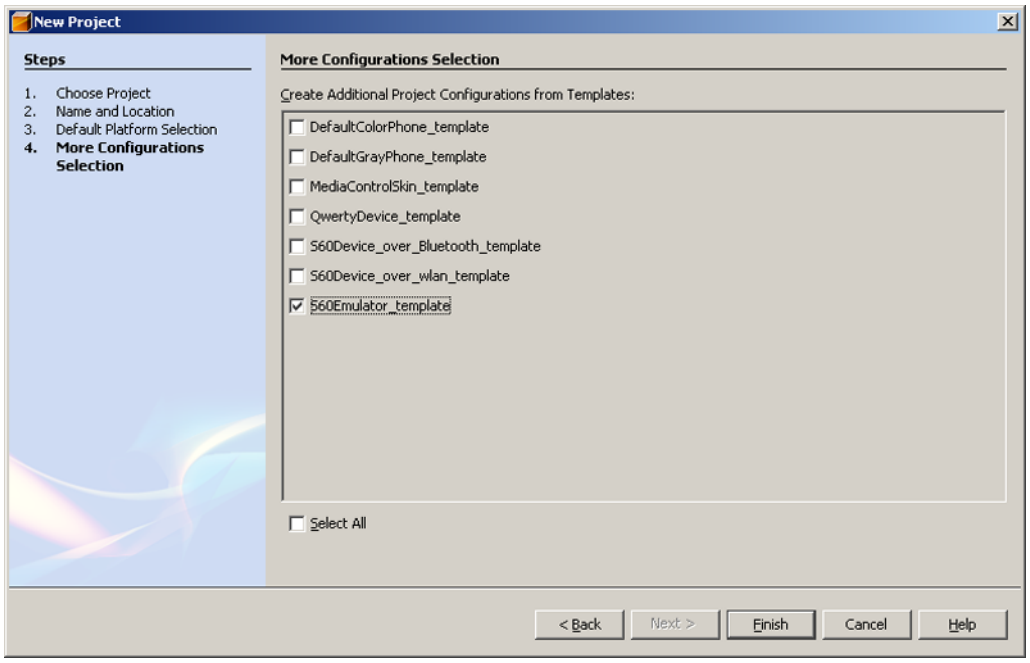


Figure 89: More cconfiguration options

9 Select S60Emulator as the template and click **Finish**.

.

Results

NetBeans creates the new project according to your definitions.

3.2.4 Using the SDK with IBM® Websphere Studio Device Developer (WSDD)

3.2.4.1 Configuring the S60 emulator as a UEI emulator device in WSDD

Context

After installing the WSDD, you need to configure the S60 emulator as a UEI (Unified Emulator Interface) emulator device in order to be able to run MIDlets on it from the WSDD.

Steps

1 **Select Devices > Configure...** or click the **Devices** icon the **WSDD** toolbar.

. The **Device Configurations** window opens:

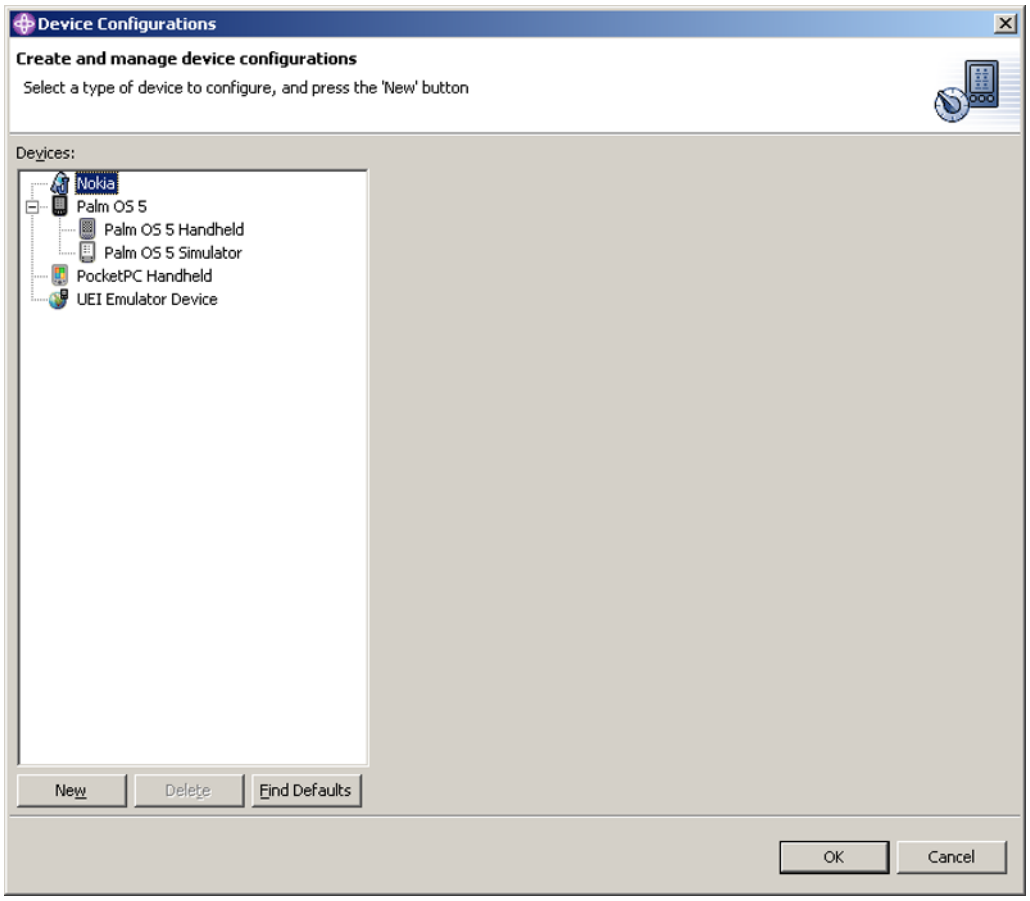


Figure 90: New UEI emulator device

2. Select **UEI Emulator Device** from the list and click **New**.

3. In the **Device Configurations** dialog, enter a name for the device by editing the **Device name** field.

4. To complete the Emulator root directory field, click **Browse** to search for the file system and select the installed S60 MIDP SDK main directory (`C:\S60\devices\S60_3rd_MIDP_SDK_FP1\`).

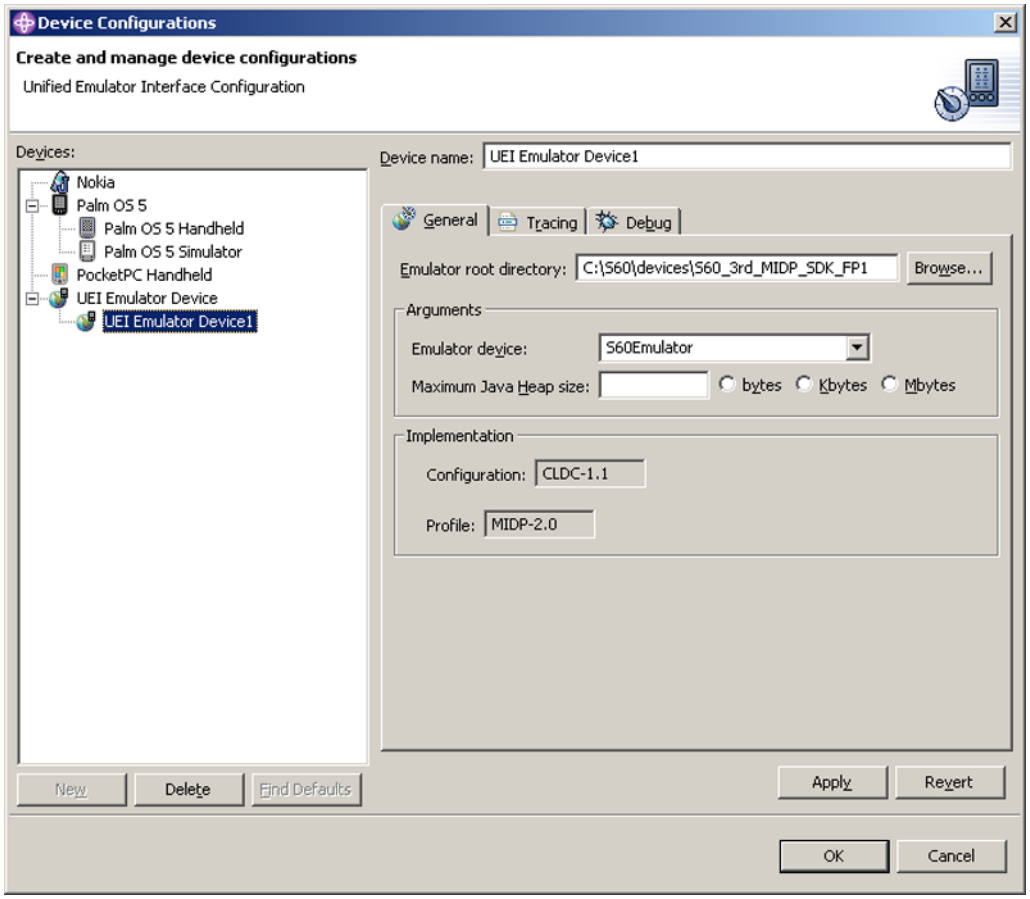


Figure 91: Configured S60 MIDP SDK device

5. Click **OK**.

Results

The S60 emulator has now been configured for WSDD, enabling you to launch applications in the emulator from WSDD.

3.2.4.2 Creating a new MIDlet suite with the WSDD

Context

One or more MIDlets can be packaged into a MIDlet suite. To create a MIDlet suite in WSDD, do the following:

Steps

1. Select **File > New > Project** from the WSDD menu bar.
2. The **New Project - Select** view is displayed:

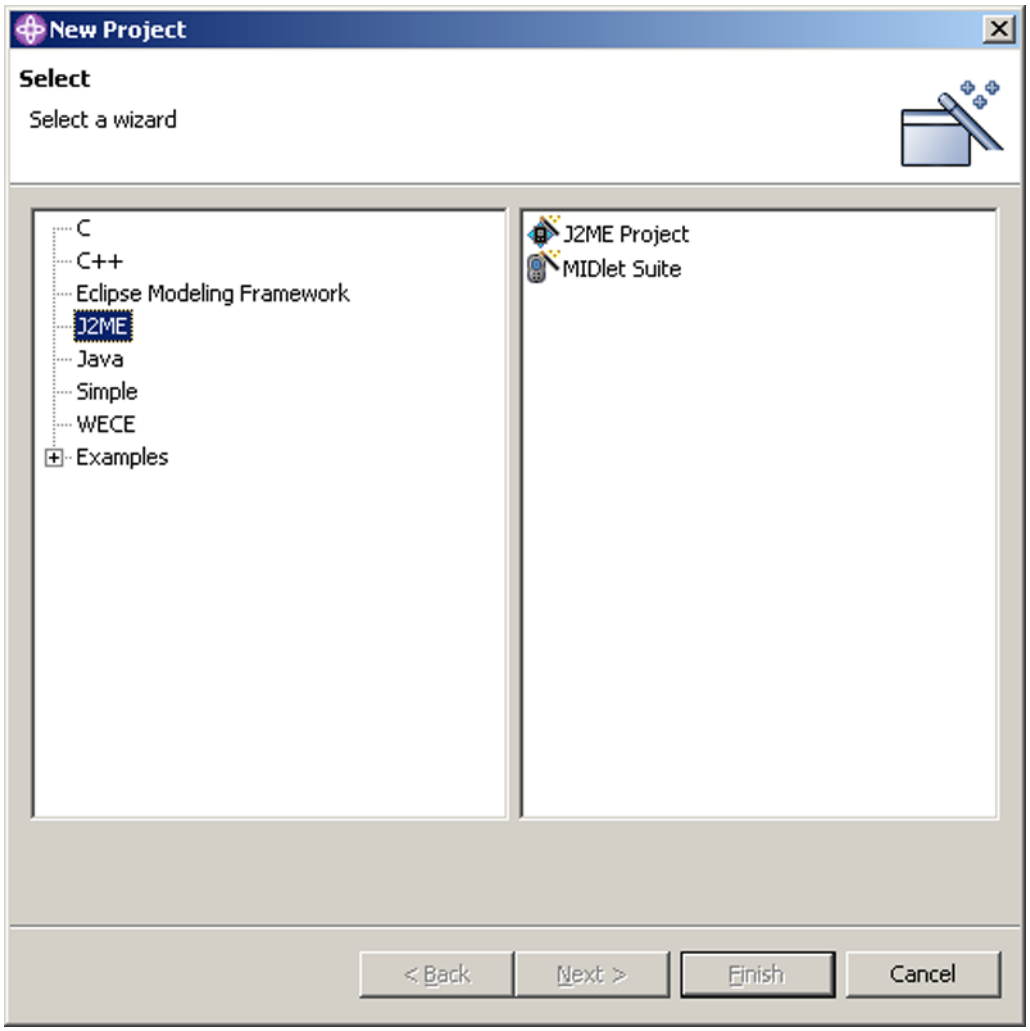


Figure 92: New Project - Select view

2 Select **J2ME** and **MIDlet Suite**.

.

3 Click **Next**.

. The **MIDlet Suite Creation** view is displayed.

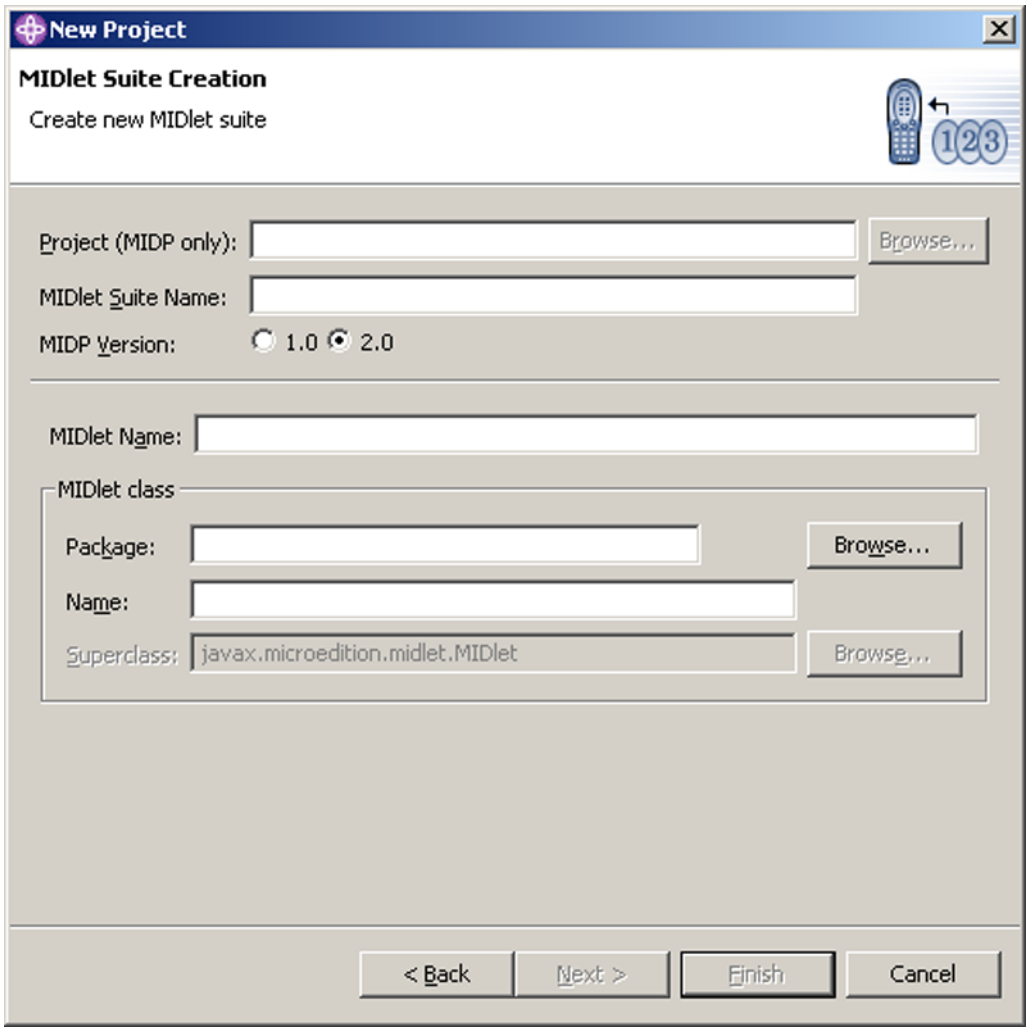


Figure 93: MIDlet Suite Creation view

4 In the **MIDlet Suite Creation** view, enter at least the following information:

. i A project name in the **Project (MIDP only)** field.

ii A name for the MIDlet suite you are creating in the **MIDlet Suite Name** field.

iii Select the MIDP version by clicking the appropriate radio button (MIDP 2.0 is recommended).

iv A name for the MIDlet that you are creating in the **MIDlet Name** field.

v A class name in the **Name** field under **MIDlet Class**.

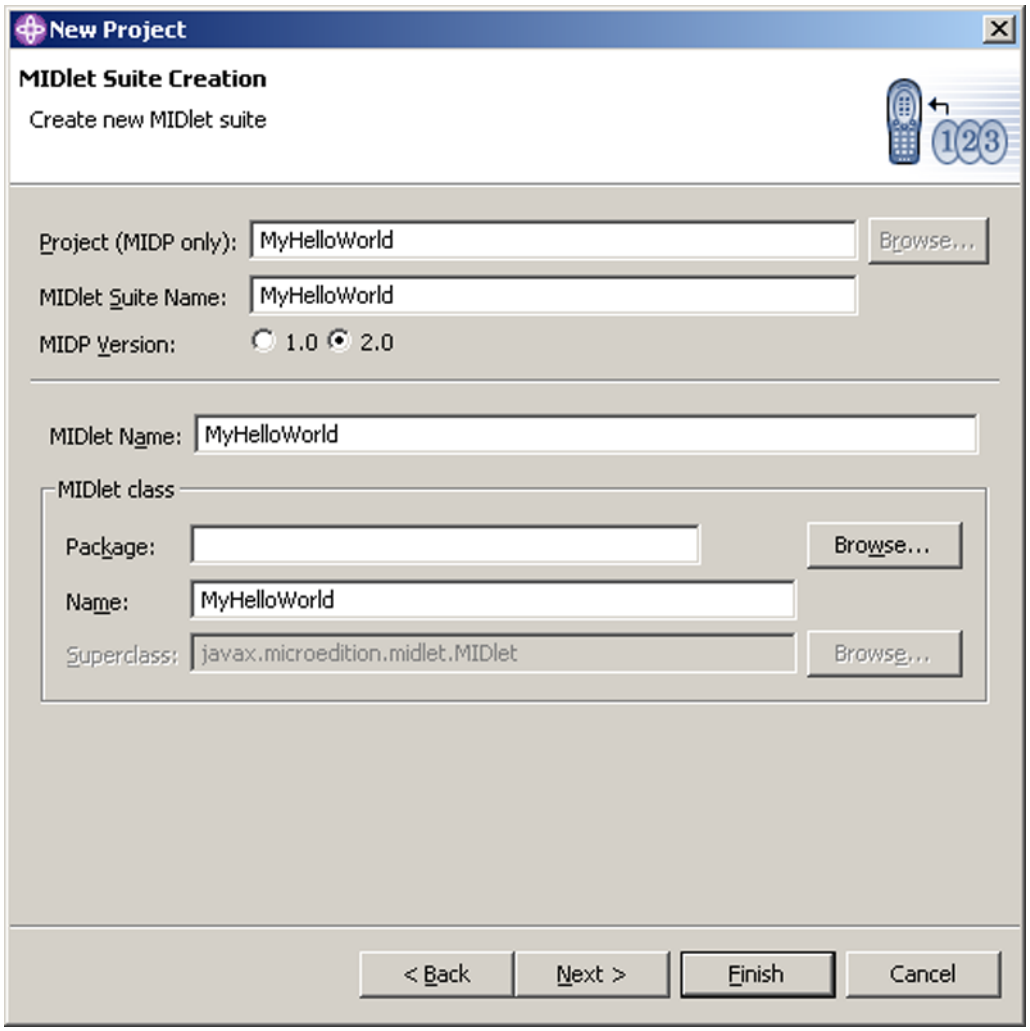


Figure 94: MyHelloWorld MIDlet data in MIDlet Suite Creation view

5. once you have entered the needed information in the **MIDlet Suite Creation** view, click **Finish**.

Results

WSDD creates a MIDlet based on the information you entered in the **MIDlet Suite Creation** view. As such, the MIDlet you have just created does not contain any content. The Importing S60 example applications to a MIDlet Suite in WSDD topic explains how edit the MIDlet by importing S60 example application content to it.

3.2.4.3 Importing S60 example MIDlets to a MIDlet Suite in WSDD

Context

Once you have created a MIDlet suite in WSDD, you can import an S60 example MIDlet into the MIDlet suite. Importing S60 example MIDlet files into WSDD is easy to do with the **Import** wizard, as described below.

Note: In order to import an S60 example MIDlet to WSDD you first need to create a MIDlet suite into which the example applicaiton source files can be imported.

Steps

1. From the WSDD main menu bar, select **File > Import**.
2. The Import wizard opens:

Figure 95: WSDD Import dialog

2. Select **File System** and click **Next**.

   The **Import from File System** dialog opens.

3. Click the **Browse** button to select the directories from which the files should be added.

4. In the **Import from directory** dialog, select the `src` folder of the S60 example application whose source files you want to import (for example, `C:\S60\devices\S60_3rd_MIDP_SDK_FP1\S60examples\helloworldplus\src`) and click **OK**.

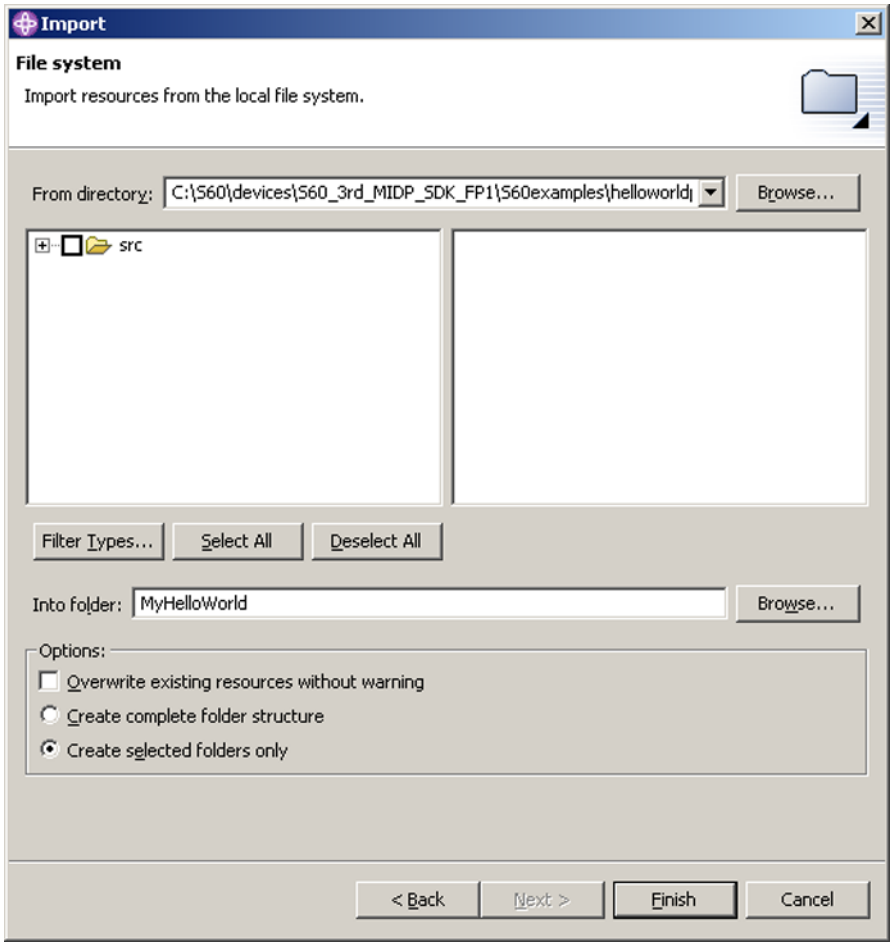


Figure 96: Importing HelloWorldPlus example application src files to WSDD

5. Select the files you want to add.

   Expand the hierarchies in the left pane and select or clear the checkboxes that represent the folders in the selected directory. Then in the right pane, select or clear checkboxes for individual files.

6. When you have finished specifying your import options, click **Finish**.

Results

The selected source files are imported ito the MIDlet suite project in WSDD:

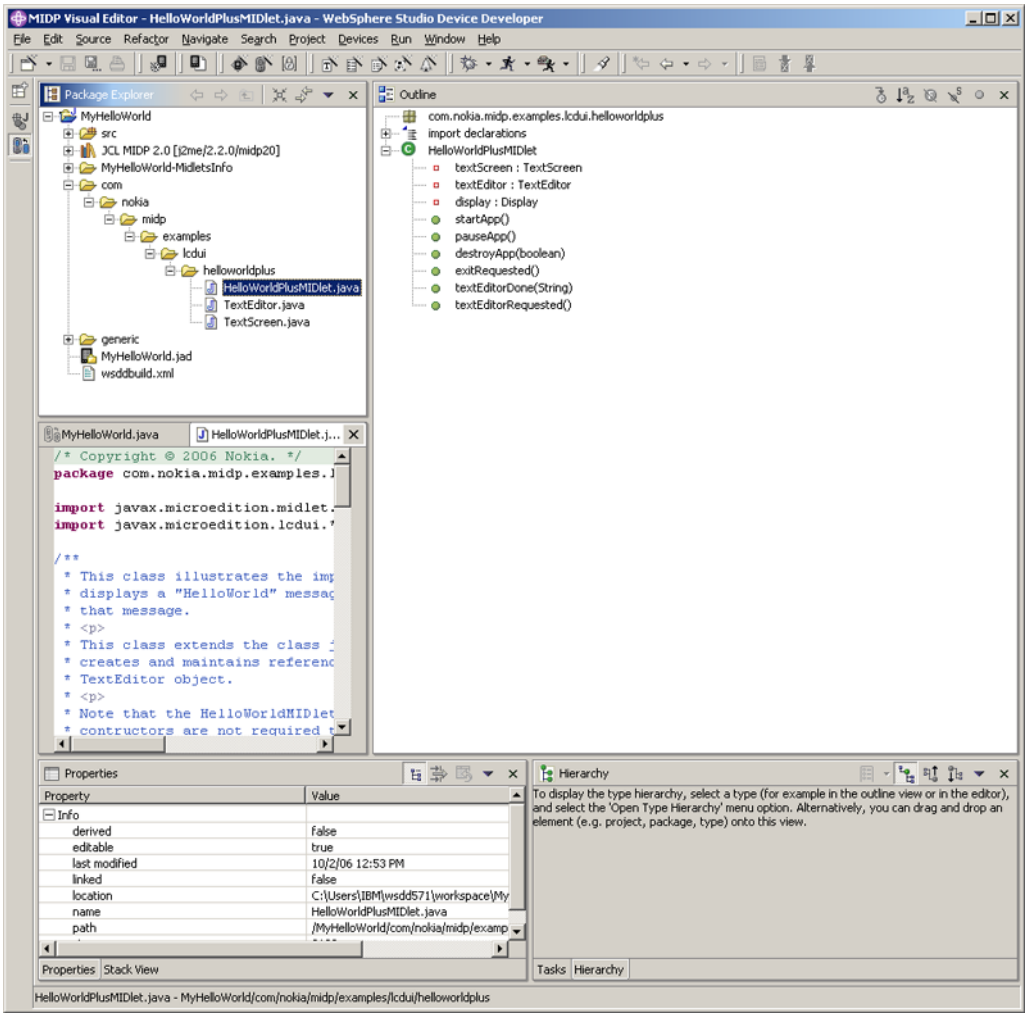


Figure 97: Imported src files of the HelloWorldPlus example application

You can use these source files to create a working MIDlet application.

3.2.4.4 Launching a MIDlet in the S60 emulator from WSDD

Context

You must create a launch configuration in WSDD the first time you run a MIDlet suite on the S60 Emulator. The executable will run on the local platform with the J9 virtual machine and the S60 Emulator.

Steps

1. Open the MIDlet project that you want to run in WSDD.
2. Select **Run > Run...** from the WSDD menu bar.

   The **Create, manage, and run configurations** dialog opens.
3. In the**Create, manage, and run configurations** dialog, select MIDlet suite from the list and click **New**.

   A dialog to configure a new MIDlet suite opens.
4. in the **Run** dialog, edit the run configuration fields.

   To fill in the **Project** field, click **Browse** to select the project. Select the configured “S60 MIDP SDK Emulator” in the **Device or JRE** field.

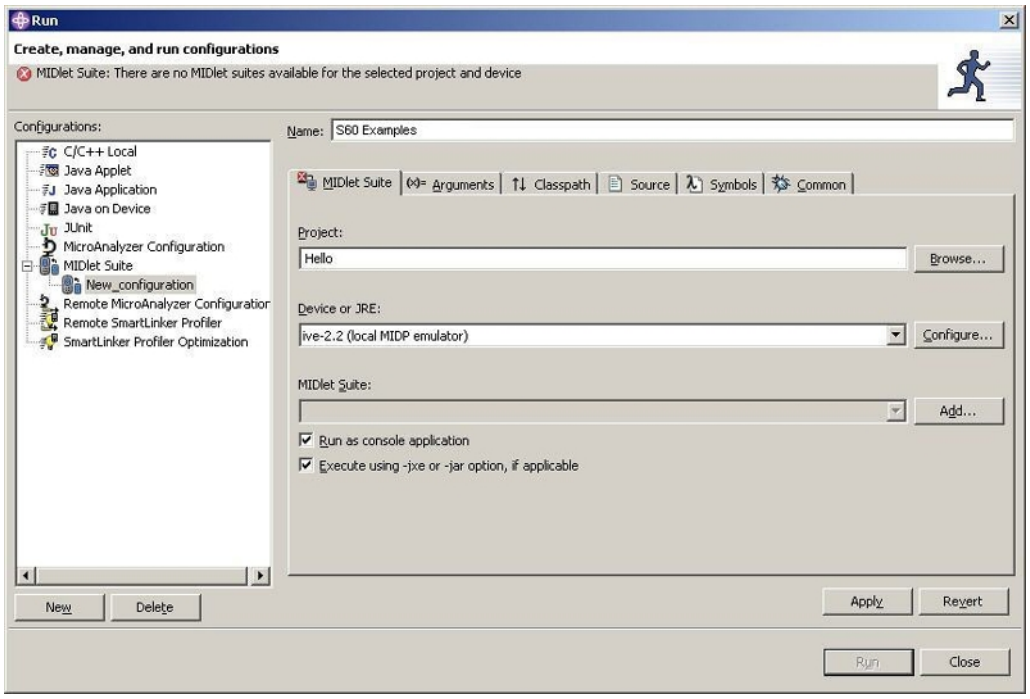


Figure 98: New MIDlet suite configuration

5 Select the MIDlet suite `examples.jad` to be run using the combo box.

.

6 Click **Run**.

. A MIDlet selector dialog is shown.

7 In the MIDlet selector dialog, select the MIDlet to run and press **Enter**.

. The J9 VM and the MIDlet launches on the S60 MIDP SDK Emulator.

	Note: If your MIDlet does not respond to commands, you can kill your MIDlet using the key combination **Shift + Ctrl + Alt + K**, which kills the application with the current keyboard focus.

You can reuse a previously created run configuration to run your MIDlet suite:

a Click **Run > Run**.

b In the **Run** dialog box, select a run configuration from the **Configurations** list.

c Click **Run**.

3.3 On-device Debugging

3.3.1 On-device MIDlet debugging over Bluetooth

	Note: In these instructions it is assumed that the Development environment has been set up correctly; to verify this please refer to Setting up the development environment.

	Note: In these instructions it is also assumed that the debug agent `<S60_SDK_installation_directory>\S60Tools\Ecmt\EcmtAgent_MIDP.SIS` has been installed on the device. The SIS file installs to native S60 applications, EcmtAgent and DebugAgent, and unpacks the `RedirMIDlet.jar` that can be used to redirect System.out and System.err streams from a MIDlet to the diagnostics view on PC. For more information about stream redirection, see Setting up system.out and system.err redirection.

For Bluetooth support, there are two supported Bluetooth drivers, Microsoft and WIDCOMM. These drivers differ slightly in the workflow and have separate instructions indicated in the following steps, when necessary.

1 In IDE device settings, set up **S60Device_over_Bluetooth** as the device to use. The location for defining this depends on the IDE in question:

   - In Eclipse, go to **Run > Debug...** dialog by editing Run/Debug configurations (Nokia SDK plugin panel).

2 Turn the device's BT radio on from the device and pair with the PC.

3 Start the EcmtAgent application on the device.

4 Discover the device's BT services from the PC.

   - If you are using the WIDCOMM driver, connect to BT com port service provided by the EcmtAgent and write down the COM port used.

5 Run the Device Connectivity Tool (**Start > Programs > S60 Developer Tools > 3rd Edition SDK > MIDP > Tools > Device Connection** or launch the exe from `<S60_SDK_installation_directory>\bin\epoc32\tools\ecmt\EcmtGw.exe`).

6 Select the correct connection method and press the **Connect** button.

   - With WIDCOMM driver, select the COM port from step 4.
   - With Microsoft driver, select the EcmtAgent Bluetooth service from the list.

7 Once the connection has been established, run the debug session in the IDE.

3.3.2 On-device MIDlet debugging over WLAN

	**Note:** In these instructions it is assumed that the development environment has been set up correctly; to verify this please refer to Setting up the development environment.

	**Note:** In these instructions it is also assumed that the debug agent `<S60_SDK_installation_directory>\S60Tools\Ecmt\EcmtAgent_MIDP.SIS` has been installed on the device. The SIS file installs to native S60 applications, EcmtAgent and DebugAgent, and unpacks the `RedirMIDlet.jar` that can be used to redirect System.out and System.err streams from a MIDlet to the diagnostics view on PC. For more information about stream redirection, see Setting up system.out and system.err redirection.

For Wireless LAN you can use both AdHoc and Infrastructure modes for On-device debugging a MIDlet.

1 In IDE device settings, set up **S60Device_over_wlan** as the device to use.

   The location for defining this depends on the IDE in question (see On-device MIDlet debugging over Bluetooth, step 1 on page 132).

2 Set up WLAN so that the PC and the device can connect to each other via WLAN network. Configure SDK preferences from the IDE to listen to a port.

3 Determine the IP address of the PC, for example using ipconfig command line tool or **Start > Settings > Network Connections > Wireless Network Connection** and the IP-address is shown on the left sidebar under **Details**.

4 Start the DebugAgent on the device and select **Options > Settings**.

5 Set the IP address or host name and port number of the PC running the IDE.

6 Start Debug Agent Server by selecting **Options > Start**.

7 Launch the debug session from the IDE.

8 Once the **Emulator Progress** dialog stops at “Connect to Agent”, select **Options > Connect** on the device.

DebugAgent

The DebugAgent is an S60 application used for On-Device Debugging over a WLAN connection. It establishes the connection between an IDE and the Java ME virtual machine on the device. The following DebugAgent settings can be changed from **Options > Settings**:

Setting	Description
Auto cleanup	If enabled, the on-screen log is cleared for each session.
Log to file	If enabled, logs are written to c:\midp2\agent.log on the device
PC host	Hostname or IP-address (preferred) of the PC running the IDE.
PC port	IDE port number for emulator WLAN connectivity, see MIDP Debugging tab in emulator preferences (via emulator or IDE).
Access Point	Asked for each connection initialization, or defined for.
Keep-alive timer	How frequently to ping the PC to keep the WLAN connection open and active. Disabling Keep-alive timer may slow down debug session initialization.
Font Size	The font size used in the log window on the device.

Connection device to PC over WLAN

For on device debugging you can use both Infrastructure and AdHoc modes for WLAN connectivity.

For Infrastructure mode you must make sure that the WLAN connectivity is not blocked between the device and the PC. Follow the instruction in the device user’s guide to set up the access point for Infrastructure mode connectivity.

Direct connection between a WLAN capable PC and the device is recommended.

To configure the device for AdHoc WLAN connectivity, proceed as follows:

1 In the device, go to **Menu > Settings > Connection > Access Points > Options** dialog.

2 Select “New access point”.

3 Specify the following parameters:

Data bearer	Wireless LAN
WLAN network name	AdHocNet
Hide WLAN network	No
WLAN network mode	Ad-hoc
WLAN security mode	Open Network
Homepage	None

4 From the **Options** menu, select **Advanced Options** and set the following parameters:

Ipv4 settings	Specify IP of the device (private IP address space is recommended, for example 192.168.0.10)

AdHoc channel	Automatic
Proxy Server address	None
Proxy port number	0
WLAN security mode	Open Network
Homepage	None

5 To configure the PC to use AdHoc WLAN connectivity, see the user's guide of your WLAN card and set it to use an IP address in the same network as the device (for example 192.168.0.11). Use this IP address to configure the DebugAgent on the device.

3.4 Device Connectivity Tool for S60 SDK

3.4.1 Installation

To use the Device Connectivity Tool, you must first install the following SIS package on your S60 smartphone: `\S60Tools\EcmtAgent_CPP.SIS`

The SIS package can be installed on the phone with, for example, OBEX file transfer, which transfers SIS packages to the phone. This will create new incoming text messages that allow you to install the SIS packages on the phone.

Note: The installation procedure may differ between phones. Please refer to the phone manual for information.

3.4.2 Server startup

Once the SIS packages have been installed, the Ecmt Server can be started. Locate the EcmtAgent application on your phone and launch it. This will start up the Ecmt Server. If this is the first time you run EcmtAgent on the device, Ecmt Server will enable a Bluetooth serial port connection between smartphone and PC, otherwise the previously used communication method will be used.

The EcmtAgent application initially displays connectivity status log to the user.

The Options menu provides the following options:

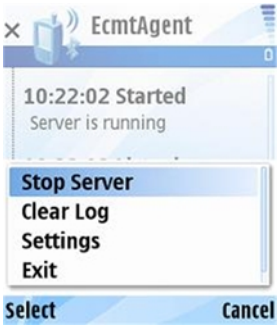


Figure 99: EcmtAgent menu options

An appropriate connectivity method can be selected through the Settings option, as demonstrated in the following figure (using the WLAN method as an example):

Figure 100: Selecting connectivity methods

Notice, that Bluetooth, WLAN and USB connectivity methods are not available in all devices. Unsupported methods are disabled in settings. When using WLAN between a smartphone and PC, a WLAN access point must exist on the smartphone, in order for it to be listed in the Select WLAN IAP query. Once you select WLAN, access point and return back to the initial view, EcmtAgent connects to the WLAN network defined by the selected WLAN IAP and displays the smartphone IP and port number which it is listening. You need to enter this information in the Device Connectivity Tool on the PC.

1 Select Tools > WLAN Settings... from the menu bar

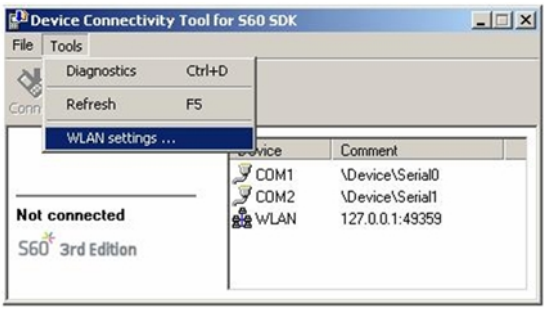


Figure 101: Defining WLAN settings in the Device Connectivity Tool

2 Enter the smartphone IP in the Handset IP field and click OK:

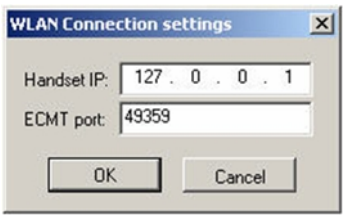


Figure 102: WLAN connection settings in the Device Connectivity Tool

USB connectivity method requires that the USB cable is connected before returning back from Settings to the initial view.

3.4.3 Client startup

Follow these steps to start up the client:

1 From the Start menu, select **Start > Programs > S60 Developer Tools > 3rd Edition SDK FP 1 > 1.0 > Tools > Device Connection.**

2 The Device Connectivity Tool for S60 SDK window is displayed:

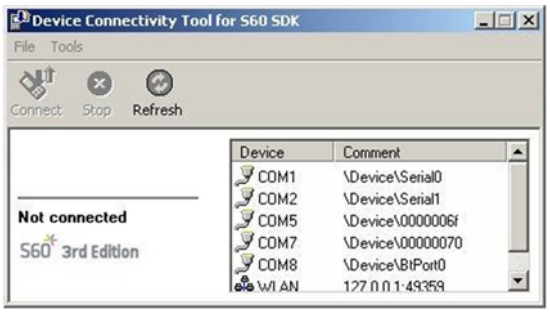


Figure 103: Device Connectivity Tool for S60 SDK - not connected

3 Use same connectivity method as EcmtAgent.

Using Bluetooth:

1 Select **Refresh**.
2 Wait until device where EcmtAgent is running, is found and listed on the port list.
3 Select the smartphone from the port list.

Using WLAN:

1 Select **Tools > WLAN Settings.**
2 Query is displayed. Enter the smartphone IP and port.
3 From the **Port** list, select the modified WLAN device.

Using USB

1 From the Port list, select the COM port which is used by the S60 smartphone on your PC.
2 Select **File > Connect**. The following window is displayed:

Figure 104: Device Connectivity Tool for S60 SDK - connected

After a connection between the PC and the smartphone has been established, select **Tools > Diagnostics** to open the Diagnostics Window.

3.4.4 System.out and System.err Redirection to PC

3.4.4.1 What is System.out and System.err redirection?

S60 3rd Edition SDK provides the possibility to view on the PC the System.out and System.err streams used by a MIDlet running on a phone with the help of the device connectivity tool. All necessary settings to enable the feature are described in Setting up System.out and System.err redirection. The ability to view System.out and System.err streams on the PC provides additional means of MIDlet debugging on a real S60 device.

System.out and System.err streams, which are written to by a MIDlet under a test, are captured by a special redirector MIDlet running simultaneously with the MIDlet being tested. The redirector MIDlet sends the data using the PC connectivity Symbian application and a Bluetooth connection to the device connectivity tool running on a PC. System.out and System.err streams are delivered to the redirector MIDlet in chunks of approximately 110 symbols. Therefore, shorter strings are not immediately seen in

the Diagnostics window, but are held in the buffer. Right after start-up the redirector MIDlet sends a “Redirector MIDlet: redirection is started” message to the diagnostics window.

The figure below shows the Diagnostics window of the device connectivity tool running on a PC. System.err and System.out printouts are seen in the SystemOut part of the window.

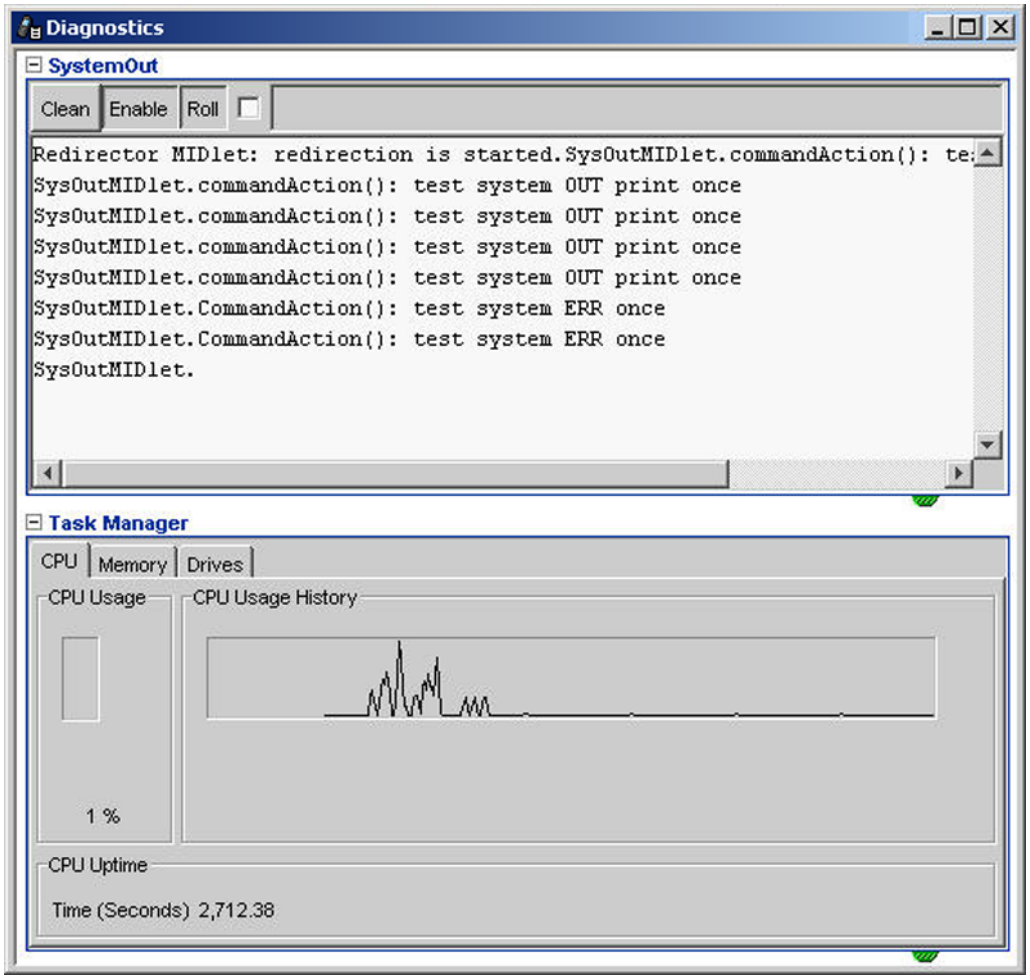


Figure 105: Device connectivity tool, Diagnostics view with System.out printouts

Related information

- Setting up System.out and System.err Redirection

3.4.4.2 Setting up system.out and system.err redirection

Steps

1. Install Ecmt applications to your S60 device from `Ecmt_Agent_MIDP.sis` file.

   See Device Connectivity Tool Installation on page 0 for instructions.
2. Install the Redirector MIDlet (`RedMIDlet`) from the Application Manager of your S60 device.

   The Redirector MIDlet is unpacked from the `.sis` file during installation, but it is not installed by default.
3. Turn on the Bluetooth connectivity on your S60 device.
4. Run the Ecmt connectivity application on your S60 device.

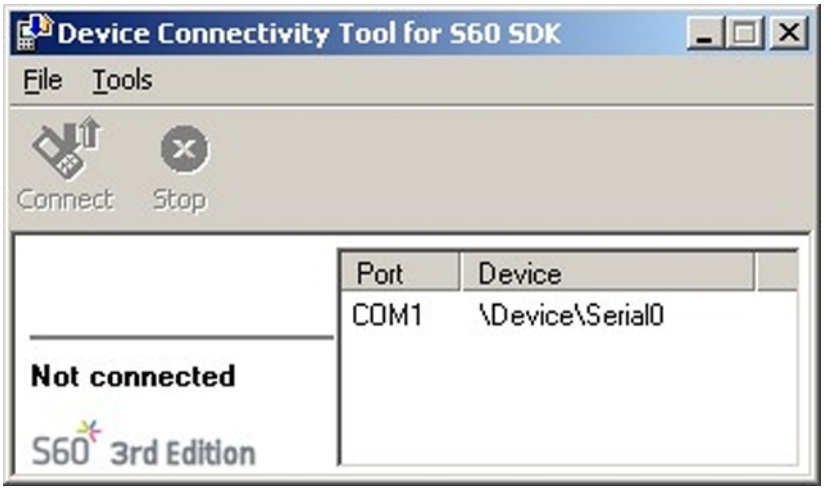
5 Set up Bluetooth connection with your S60 device as follows:

i Turn on the Bluetooth connectivity of your PC, scan and pair the S60 device, and discover available services.

ii Connect to the Bluetooth Serial port. The Bluetooth services window can be seen in the figure below.

6 Run the device connectivity tool on the PC by selecting **Start > All Programs S60 Developer Tools >3rd Edition FP1 SDK > MIDP > Tools > Device Connection**.

i In the **Port** tab of the device connectivity tool, select the COM port provided by Bluetooth connection with the phone (Bluetooth serial port).



ii Click the **Connect** button.

iii Select **Tools > Diagnostics** . The Diagnostics window appears.

7 Run the Redirector MIDlet on the phone. A message saying “Redirector MIDlet: redirection is started” should appear in the **Diagnostics** window.

8 Run the MIDlet under test. Messages sent to System.out and System.err will start to appear once the 110 symbol buffer becomes full.

3.5 Command Line Interface

3.5.1 Command line interface

The SDK has two command line interfaces:

- Emulator Command line interface (`emulator.exe`)
- SDK Command Line Interface (`sdk.exe`)

emulator.exe

Typically, an IDE that supports MIDlet development automatically uses emulator.exe when it uses the SDK as a device. A developer of MIDlets is unlikely to use emulator.exe on the command line to monitor the behavior of a MIDlet. The SDK ignores options and commands that emulator.exe does not support.

`emulator.exe` is the implementation of the Unified Emulator Interface Specification (UEI) 1.0.2.

`emulator.exe` is located in `<S60_SDK_installation_directory>\bin\`

For syntax and arguments, see Emulator.exe syntax and arguments.

sdk.exe

The executable `sdk.exe` supports options and commands that you use in message content development. If you are developing message or browser content, you are more likely to use `sdk.exe` on the command line because with this executable you can work within a shell and arrange to have one instance of the SDK send a message to another instance. Options and commands that `sdk.exe` does not support cause the entire command to fail, and an error message to be posted.

`sdk.exe` is located in `<S60_SDK_installation_directory>\bin\epoc32\release\winscw\udeb`.

The **SDK command line interface** (`sdk.exe`) enables web content developers to easily test their content on the emulator or on a device. This interface can be used to:

- Launch the emulator, view particular content in an already running emulator (for example an image) or launch a URL in the emulator.
- Send a URL to a mobile device and launch the browser with the URL. This command line interface relies on the Device connectivity tool for S60 SDK on page 0 to provide connectivity between the PC and the mobile device.

`sdk.exe` will automatically launch the Device Connectivity Tool for S60 SDK tool - if it is not already running - for all the operation is in the device.

For syntax and arguments, see Sdk.exe syntax and arguments.

3.5.2 Emulator.exe syntax and arguments

`emulator.exe` is the implementation of the Unified Emulator Interface Specification (UEI) 1.0.2.

Syntax

```
emulator [arguments] <Application>
```

Note: The following directory path must always be added when the `emulator.exe` command is entered into the command prompt:

`C:\<S60_SDK_installation_directory>\bin\`

Arguments

`-classpath, -cp`	Path to jar file and MIDlet class
`-D<property=value>`	Property definitions.
`-version`	version information of the emulator, profile and configuration.
`-help`	Displays a list of valid arguments.

`-Xverbose[:allocation | gc | gcverbose | class | classverbose | verifier | stackmaps | bytecodes | methods | methodsverbose | frames | stackchunks | exceptions | events | threading | monitors | networking | all ]`	Enables verbose output.
`-Xquery`	Displays emulator device information.
`-Xdebug`	Use a remote debugger (Device Connectivity Tool).
`-Xrunjdwp: [transport=<transport>, address=<address>,server=[y|n],suspend=[y|n]`	Debugging options.
`-Xdescriptor:<jad file name>`	The JAD file to be executed.
`-Xheapsize:<size> (e.g. 65536, 128k or 1M)`	Specifies the VM heapsize (overrides default value)

Example

(Enter the following as an one-line command.)

```
C:\<SDK_installation_directory>\bin\emulator -
Xdevice:<SDK_installation_directory> -Xdescriptor:c:\myprojects
\mymidlet.jad
```

3.5.3 Sdk.exe syntax and arguments

Syntax

```
sdk.exe [argument]
```

	**Note:** The full directory path must always be added when the `sdk.exe` command is entered into the command prompt. For example: `C:\<S60_SDK_installation_directory>\bin\epoc32 \release\winscw\udebsdk.exe`

Arguments

`load <url/filename>`	Starts the emulator if it is not running and loads the specified URL/Filename into the S60 browser.
`-shell`	Starts an interactive session, which issues a prompt, and read commands from the standard input.
`target:<target>`	Defines the target for the operation. Valid target arguments are `emulator` and `device`. The default value for this argument is `emulator` (that is, if no target is defined the operation will be executed in the emulator). `sdk.exe` will attach to an already running emulator executable and attempt to load the content in it.

`-com:<com_port>`	Defines the communication port (serial port) to communicate with a device. This option is only used when the target is the device. The value for the `<com_port>` argument value is a positive integer representing the port number. If the Device Connectivity Tool for S60 SDK application is already running, this option will reconfigure the application and it will force the Device Connectivity Tool for S60 SDK application to use the newly defined port. This option is optional if the Device Connectivity Tool for S60 SDK application is already running and configured.
`-url:<url/pathname>`	This option specifies the URL or a path for a content. The URL can be of the `http://`, `https://` or `file://` schema. The `-url:` prefix for this argument is optional and it is only supported for backward compatibility with some 3rd party applications which use this interface to pass content to the SDK. A file path or a file schema is always relative to the hardware (PC or mobile device) where the content is rendered (launched). If the target is the emulator, file references are relative to the PC. If the target is a device, file references are relative to the device.
`close [-exit]`	The `close` command terminates the session but leaves the SDK running, while `close -exit` terminates the session as well as the emulator.

Limitations

This command line interface will not transfer files to the device. To open a local file on the device from the PC using this CLI, the content should be transferred to the device using other tools.

Examples

```
C:\<S60_SDK_installation_directory>\bin\epoc32\release\winscw
\udeb>sdk.exe http://www.s60.com/
```

- Opens the http://www.s60.com/ web page in the emulator Web browser. This will start the emulator if the emulator is not running, or it will attach to an already running emulator.

```
C:\<S60_SDK_installation_directory>\bin\epoc32\release\winscw
\udeb>sdk.exe C:\mypage.html
```

- Loads the C:\mypage.html page to the emulator's Web browser from the root directory of the PC's C: drive.

```
C:\<S60_SDK_installation_directory>\bin\epoc32\release\winscw
\udeb>sdk.exe -target:device -com:2 C:\mypage.html
```

- Loads the C:\mypage.html page to the Web browser on the device from the root directory of the `C:` drive of the device. Notice, that on the device some directories are protected and the browser application does not have access rights to those directories (the `C:\` directory is not protected).

3.6 Tools for Location-Based Applications

3.6.1 Tools used for developing location-based applications

The S60 SDK provides a number of APIs which can be used for developing location-based applications for devices that provide support for location acquisition.

To support location-based application development, the S60 SDK includes tools which can be used to fully test these applications in the emulator. The figure below illustrates the relation of different tools and the location framework that are available for application developers.

	Note: The code samples provided in the instructions in this section are C ++ examples. For corresponding MIDP code, please refer to Location API (JSR-179) provided in the Java API Documentation section of the SDK Help.

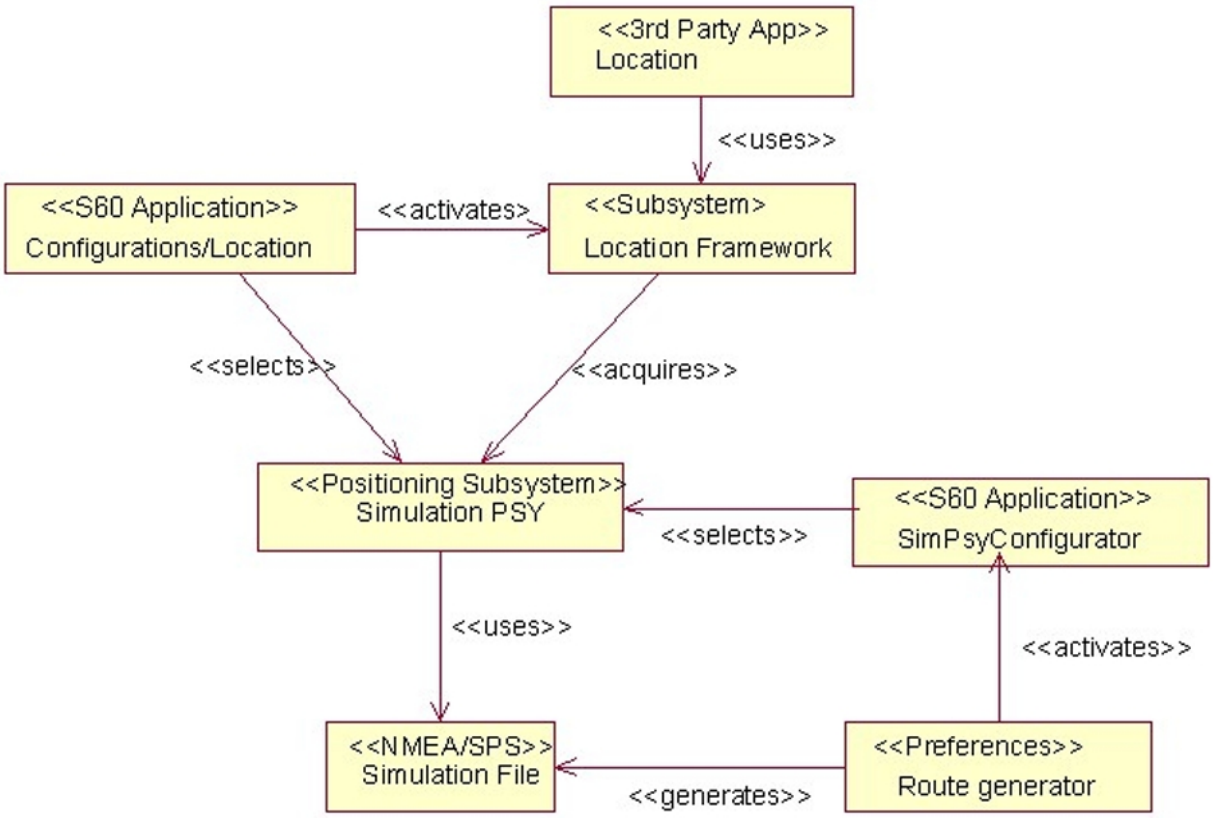


Figure 106: Relation of different tools and the location framework available for application developers

Location Application

Application developed using location APIs.

Location Framework

A framework that provides location-based services and location APIs for application developers. For more details on the location framework, see the API Reference Guide.

Configurations/Location

Simulation PSY

An application in the emulator that is mainly used to activate the Location functionality and to select the desired positioning subsystem (PSY). The positioning subsystem provided in the emulator is the Simulation PSY.

A positioning subsystem used in the emulator.

Note: You can also use the Simulation PSY in case you wish to test your application using simulated locations in the target device. This requires that the SimPsyConfigurator is installed on the device.

SimPsyConfigurator

A tool primarily used to select the desired simulation file used by the Simulation PSY. The SimPsyConfigurator is used for selecting the desired simulation file format: NMEA, or SPS format. For more details, see the Simulation PSY Configurator user's guide.

Note: SimPsyConfigurator can also be installed into a target device. Install the SimPsyConfigurator.sis (included with the SDK) file to your device, which has the required location support.

Route Generator

The Route Generator is used to generate new simulation files used by the Simulation PSY. This tool is used via the emulator Preferences Window. For more details, see the Simulation PSY Route Plug-In user's guide.

Simulation File

A file in either NMEA or SPS format which is used by the Simulation PSY.

3.6.2 Simulation PSY user's guide

3.6.2.1 Introduction

This document describes the usage of Simulation PSY. Simulation PSY is available in S60 platform 2.6 and onwards

A developer of a location based application needs to retrieve positions when developing the application. With Simulation PSY it is possible to get positions when running applications in the Windows emulator environment and on target device.

When testing, Simulation PSY enables simulation of different positions without actually needing to go to these different locations. With this software, the retrieval of positioning data is independent of any hardware.

Simulation PSY makes it easy to reproduce different locations, for example, to simulate positions on the North Pole, or to test how an application behaves when moving over the Zero Meridian. Without Simulation PSY it is hard to test applications in this kind of situations. It is useful to test how a location application behaves when changing the accuracy of the location estimates. This is possible with Simulation PSY. It is also possible to test error situations.

Simulation PSY reads location data from a file and returns location estimates to the location based application. It is possible to get the same position all the time or simulate movement in a specified direction.

Simulation PSY can be used in three different modes:

- NMEA mode; reads NMEA sentences from a file.
- Simulated Movement mode; reads simulated movement data from a file.
- Fixed Data mode, where it returns a fixed set of location information.

The mode in which Simulation PSY runs depends on the file extension that the simulation file setting ends with.

The file extension NME means that NMEA mode is used. For the behavior of this mode, see NMEA File Format in File formats.

The file extension SPS means that Simulated Movement mode is used. For the behavior of this mode, see Simulated Movement Format in File formats.

If the simulation file setting is not configured then Simulation PSY returns location information from a predefined default NMEA file. If the default NMEA file is also missing then it runs in fixed data mode where it returns fixed basic location information consisting of the following items:

Position data	Value
Latitude	0
Longitude	0
Altitude (WGS 84)	0
Horizontal Accuracy	0
Vertical Accuracy	0
Datum	WGS-84
Time	Current System Time

3.6.2.2 Configuration

Simulation PSY returns location information from a default NMEA file if there is no configuration. In case the default NMEA file is missing then fixed location information is returned as specified in the previous section.

Simulation PSY needs to be configured for simulating specific locations. Configuration defines which simulation file Simulation PSY will use. Configuration is made with the Simulation PSY Configurator application. For further information, see S60 Simulation PSY Configurator User Guide.

3.6.2.3 Connecting and retrieving a location

For information on how to retrieve locations from the Location Framework, see S60 Location Acquisition API Specification.

To get a location from the Simulation PSY, a user first needs to connect to `RPositionServer`. Then it is possible to open a sub-session to Simulation PSY using `RPositioner` via `RPositionServer`. Before opening a sub-session to Simulation PSY, the PSY should be enabled in the S60 platform Location Settings UI.

For example:

```
TPositionInfo posInfo;
TRequestStatus status;
RPositionServer positionServer;
RPositioner simPSY;

User::LeaveIfError(positionServer.Connect());
CleanupClosePushL(positionServer);
User::LeaveIfError(simPSY.Open(positionServer));
CleanupClosePushL(simPSY);

// Set Requestor information
User::LeaveIfError(simPSY.SetRequestor(CRequestor::ERequestorService,
CRequestor::EFormatUrl, _L("http://www.example.com")));

simPSY.NotifyPositionUpdate(posInfo, status);
User::WaitForRequest(status);

// Do something with posInfo

CleanupStack::PopAndDestroy(2, &positionServer); // simPSY
```

Alternatively, the sub-session to Simulation PSY can be opened directly using its positioning plug-in ID, 0x101f7a81.

Simulation PSY supports the class types `TPositionInfo`, `TPositionCourseInfo`, `TPositionSatelliteInfo` and `HPositionGenericInfo`, which means that the argument to `RPositioner::NotifyPositionUpdate()` must be of these classes. `HPositionGenericInfo`, `TPositionCourseInfo` and `TPositionSatelliteInfo` are all derived from `TPositionInfo`.

In the Simulated Movement mode, no class specific position information is stored in the incoming position class. Only position information specific to `TPositionInfo` is stored.

In the NMEA mode, position data is stored depending on the incoming position class. For `HPositionGenericInfo`, a request field is present for each retrievable position data. If a field is set as requested in `HPositionGenericInfo`, the Simulation PSY stores the corresponding position data. For example, if the field `EPositionFieldNMEASentences` is set as requested, the raw NMEA sentences that constitute the position fix are stored in `HPositionGenericInfo`. The number of the returned NMEA sentences is placed into the `EPositionFieldNMEASentences` field. The returned type is TUint8. The first sentence is placed into the `EPositionFieldNMEASentencesStart` field, the second sentence in the `EPositionFieldNMEASentencesStart+1` field, etc. The returned type is `TDesC8`. The format of the returned location estimates, WGS 84, is used in both NMEA and Simulated Movement mode.

In Fixed Data Mode, no class specific position information is stored in the incoming position class. Only position information specific to `TPositionInfo` is stored.

When two clients connect to Simulation PSY, two instances are constructed. It is possible for both to read from the configuration file and/or simulation file at the same time. The outputs of the two sessions are completely independent of each other.

It is possible to set update options on a positioning plug-in in the Location Acquisition API (see S60 Location Acquisition API Specification). Simulation PSY supports the following update options: timeout, interval and partial updates. Partial updates are only supported in the NMEA mode, meaning that no partial updates could be returned for the Simulated Movement mode. See S60 Location Acquisition API Specification for more information about partial updates.

`HPositionGenericInfo`, `TPositionCourseInfo` and `TPositionSatelliteInfo` are supported in both modes. Only the NMEA mode supports position data specific for these position classes.

Error situations

When trying to open the PSY with `RPositioner::Open()`, one of these error codes could be returned:

- `KErrNotFound`; Simulation PSY cannot find a needed file.
- `KErrCorrupt`; the syntax of the Simulated Movement file is incorrect or all words are not present.
- `KErrNotSupported`; the extension of the simulation file is recognized, neither of SPS nor NME.

When trying to get a location estimate with `RPositioner::NotifyPositionUpdate()`, one of these error codes could be returned:

- `KPositionQualityLoss`; the configuration for requests is set to fail in the Simulated Movement mode.
- `KPositionPartialUpdate`; NMEA is missing position info and the client allows partial updates.

- System wide error codes from `TLexer` and `Math`
- Other system wide error codes.

When trying to cancel a location estimate with `RPositioner::CancelNotifyPositionUpdate()`, following error code could be returned:

- `KErrCancel`; ongoing request is properly cancelled.

3.6.2.4 Status reporting

For information on how to retrieve module status from the LFW, see S60 Location Acquisition API Specification.

To get module status from the Simulation PSY, a user first needs to connect to `RPositionServer` and open a sub-session to Simulation PSY using `RPositioner`. See S60 Location Acquisition API Specification.

Example of how to retrieve module status from Simulation PSY:

```
TPositionModuleStatus moduleStatus;
TPositionModuleStatusEvent moduleEvent;
TRequestStatus status;
RPositionServer positionServer;
RPositioner simPSY;
const TPositionModuleId KSimulationPSYUid = {0x101f7a81};

User::LeaveIfError(positionServer.Connect());
CleanupClosePushL(positionServer);

// Asynchronous module status
moduleEvent.SetRequestedEvents(TPositionModuleStatusEvent::EEventAll);
positionServer.NotifyModuleStatusEvent(moduleEvent, status,
KSimulationPSYUid);

TInt err = simPSY.Open(positionServer, KSimulationPSYUid);
// Variable err could be checked
CleanupClosePushL(simPSY);
User::WaitForRequest(status);

moduleEvent.GetModuleStatus(moduleStatus);
TPositionModuleStatus::TDeviceStatus deviceStatus
    = moduleStatus.DeviceStatus();
TPositionModuleStatus::TDataQualityStatus qualityStatus
    = moduleStatus.DataQualityStatus();

// Do something with moduleStatus, deviceStatus, qualityStatus

// Synchronous module status
User::LeaveIfError(positionServer.GetModuleStatus(moduleStatus,
    KSimulationPSYUid));

deviceStatus = moduleStatus.DeviceStatus();
qualityStatus = moduleStatus.DataQualityStatus();

// Do something with moduleStatus, deviceStatus, qualityStatus

CleanupStack::PopAndDestroy(2, &positionServer); // simPSY
```

Simulation PSY reports module status during its life cycle. Module status is reported only when Simulation PSY is opened and closed, i.e. no status is reported during location requests.

The following module statuses are reported:

Module Status	Event Group	Comment
`EDeviceError`	DeviceStatus	Reported if an input file cannot be found or is corrupt

`EDeviceInactive`	DeviceStatus	Reported when Simulation PSY is unloaded.
`EDeviceReady`	DeviceStatus	Reported when Simulation PSY is online and ready to retrieve position information.
`EDataQualityUnknown`	DataQualityStatus	Reported when Simulation PSY is unloaded.
`EDataQualityNormal`	DataQualityStatus	Reported when Simulation PSY is online and ready to retrieve position information.

3.6.2.5 File formats

NMEA File Format

Simulation PSY can read files that contain NMEA sentences with NMEA 0183, version 2.1.

The NMEA file is in 7-bit ASCII format (see NMEA 0183 version 2.1). All lines end with the character '\n'. The NMEA sentences are case sensitive.

Simulation PSY parses position information from the NMEA sentences GGA, RMC, GSA and GSV. Data from other sentences are not parsed. The NMEA sentences GLL, GGA, GSA, and RMC are required to make a location estimate.

Simulation PSY has to understand which sentences constitute the location estimate. When a required NMEA sentence is read for the second time, the parsing of NMEA data stops, and Simulation PSY checks if all required sentences have been read. The location estimation continues if all the required sentences have not been read.

When the end of a file is reached, Simulation PSY starts re-reading again from the beginning of the file. The parser is reset, meaning that all the required sentences need to be read again starting from the beginning of the file before a location estimate is returned.

Satellite time is read from the RMC sentence and stored in the incoming position class if it is an instance of `TPositionSatelliteInfo` or `HPositionGenericInfo`. In the latter case, satellite time is stored only if it is a requested field in the `HPositionGenericInfo` instance. System time is set in the base class `TPositionInfo` at each position reading.

The following table shows which position data, parsed from an NMEA data file, each position class supports. Basic info in the table below indicates that `TPositionInfo` supports position data. All read NMEA sentences are included in `HPositionGenericInfo` if `EPositionFieldNMEASentences` is requested.

NMEA Source	**Position data**	**LFW class type** **(Course/ satellite class tree)**	**LFW class type** **(Generic info class tree)**	**Comment**

GGA	Latitude	Basic info		
	Longitude	Basic info		
	Altitude (WGS 84)	Basic info		Calculated as mean sea level altitude plus Geoidal separation.
	Geoidal separation	--	Generic info	
	Altitude (sea level)	--	Generic info	Taken directly from GGA
RMC	Satellite time	Satellite info	Generic info	
	True course	Course info	Generic info	
	Speed	Course info	Generic info	
	Magnetic course	--	Generic info	Calculated as true course + magnetic variation.
GSA	Number of used satellites	Satellite info	Generic info	Number of satellites that are used in position fix.
	PRN (for each satellite)	Satellite info	--	
	HDOP	Satellite info	Generic info	
	VDOP	Satellite info	Generic info	
	PDOP	--	Generic info	If PDOP is not present in the sentence, it is calculated as sqrt(HDOP^2 + VDOP^2).
	Horizontal accuracy	Basic info		Calculated as HDOP * UERE where UERE is 10m.
	Vertical accuracy	Basic info		If fix is in 3D mode, it is calculated as 1.5 * HDOP * UERE where UERE is 10m.

GSV	Number of satellites in view	Satellite info	Generic info	Calculated total number of satellites from GSV sentences. In the satellite info object, this value indicates how many satellites this information is based on, Number of satellites in view cannot be set explicitly.
	Elevation	Satellite info	--	
	Azimuth	Satellite info	--	
	Signal strength	Satellite info	--	
All	Raw NMEA data	--	Generic info	

An example of correct NMEA file content:

```
$GPGSV,3,2,12,24,32,051,,22,21,292,,01,20,315,,14,20,321,*71
$GPGSV,3,3,12,05,19,134,,13,18,348,,17,14,180,,10,10,110,*7F
$GPGLL,6459.8757,N,01433.8547,E,091931.375,A*3E
$GPGGA,091931.38,6459.8757,N,01433.8547,E,1,04,2.0,-0034,M,,,,*3D
$GPRMB,A,0.21,L,SIM003,SIM001,6500.0900,N,01433.8589,E,
000.2,00..,000.0,V*18
$GPRMC,091931.38,A,6459.8757,N,01433.8547,E,21.7,177.3,200302,02.,E*6F
$GPAPB,A,A,0.2,L,N,,,87.6,M,SIM001,358.1,M,,,*16
$GPGSA,A,3,01,02,03,04,,,,,,,,1.0,2.0,3.0*34
```

Position information is obtained after parsing the GLL, GGA, RMC and GSA sentences. Latitude is parsed from “6459.8757,N”, longitude from “01433.8547,E”, altitude from “-0034”, and the value “2.0” is used for calculating horizontal and vertical accuracy. All retrieved from GGA sentence.

```
$GPGLL,6459.8757,N,01433.8547,E,091931.375,A*3E
$GPGSV,3,1,12,06,53,095,,30,52,150,,15,51,090,,25,37,230,*7F
$GPGSV,3,2,12,24,32,051,,22,21,292,,01,20,315,,14,20,321,*71
$GPGGA,091931.38,6459.8757,N,01433.8547,E,1,04,2.0,-0034,M,,,,*3D
$GPRMB,A,0.21,L,SIM003,SIM001,6500.0900,N,01433.8589,E,
000.2,00..,000.0,V*18
$GPRMC,091931.38,A,6459.8757,N,01433.8547,E,21.7,177.3,200302,02.,E*6F
$GPAPB,A,A,0.2,L,N,,,87.6,M,SIM001,358.1,M,,,*16
$GPGSA,A,3,01,02,03,04,,,,,,,,1.0,2.0,3.0*34
```

The first line in the following textbox starts with a GLL sentence which has already been parsed. The other three sentences required for calculating position information, i.e. GGA, RMC and GSA, have also been parsed. Hence a second position fix is obtained from the parsed sentences.

```
$GPGLL,6459.8757,N,01433.8547,E,091931.375,A*3E
$GPGSV,3,1,12,06,53,095,,30,52,150,,15,51,090,,25,37,230,*7F
$GPGSV,3,2,12,24,32,051,,22,21,292,,01,20,315,,14,20,321,*71
```

Rest of NMEA file.

An example of incorrect NMEA file content..

```
$GPGSV,3,2,12,24,32,051,,22,21,292,,01,20,315,,14,20,321,*71
$GPGSV,3,3,12,05,19,134,,13,18,348,,17,14,180,,10,10,110,*7F
$GPGLL,6459.8757,N,01433.8547,E,091931.375,A*3E
$GPGGA,091931.38,6459.8757,N,01433.8547,E,1,04,2.0,-0034,M,,,,*3D
```

```
$GPRMB,A,0.21,L,SIM003,SIM001,6500.0900,N,01433.8589,E,
000.2,00..,000.0,V*18
$GPRMC,091931.38,A,6459.8757,N,01433.8547,E,21.7,177.3,200302,02.,E*6F
$GPGGA,091931.38,6434.8744,N,01343.8637,E,1,04,2.0,-0032,M,,,,*3D
```

GGA sentence exists for the second time before a GSA sentence is read. Location estimation for read NMEA sentences is not used. Simulation PSY will continue reading NMEA sentences trying to read a correct fix.

The Simulation PSY supports the NMEA sentences from the following GPS devices:

- Garmin:
  - eTrex – basic
  - eTrex – Legend
  - III plus
  - 12 cx
- Magellan:
  - GPS 315
  - Meridian Gold
  - Tracker

Simulated Movement Format

Simulation PSY can generate location data from simulated information.

The Simulated Movement file has the following format:

```
Horizontal accuracy={float};
Vertical accuracy={float};
TimeToFix min={integer};
TimeToFix max={integer};
Powerup time={integer};
Longitude={float};
Latitude={float};
Speed={float};
Course={float};
[Deterministic|Random]={integer};
```

The Simulated Movement file is in 7-bit ASCII format. All lines end with the character '\n'. The Simulated Movement file words are case insensitive. The Simulated Movement file is read when a sub-session to Simulation PSY is opened.

The Simulation Movement file words before the '=' mark must have spaces as shown above. However, it is legal to have spaces before and after the '=' mark and before the semicolon (;).

All Simulated Movement sentences end with a semicolon (;).

No comments are allowed in the Simulated Movement file.

The order of different lines does not matter, but it is required that all simulated movement sentences exist in the file.

The float value decimal separator is a period (.).

Parameter	Parameter type	Value type	Value range	Examples, delimited by ;
`Horizontal accuracy`	Meter (m)	Float	0 -	23.56; 56
`Vertical accuracy`	Meter (m)	Float	0 -	30.89899; 12
`TimeToFix min`	Seconds (s)	Integer	0 -	0; 2

`TimeToFix max`	Seconds (s)	Integer	0 -	4; 5
`Powerup time`	Seconds (s)	Integer	0 -	3; 6
`Longitude`	Longitude	Float	-180 - 180	-179.686; -180; 180; 0;
`Latitude`	Latitude	Float	-90 - 90	51.568; -90; 90; 0
`Speed`	Meter per Second (m/s)	Integer	0 -	0; 5
`Course`	Degrees	Integer	0 - 360	78; 0; 360
`Deterministic | Random`	Simulation setting	Integer	0 -	0; 3; 5

If `Speed` is not 0, new locations are calculated for each location estimate.

The `Course` value is the direction of the simulated movement (0 and 360 = North, 90 = East, 180 = South and 270 = West).

The last row specifies how to simulate errors. If `Deterministic` is used, positioning fails at the specified frequency. `Deterministic=3` means that every third request fails. Random means that positioning fails sporadically, but at the specified mean frequency. `Random=3` means that after a large amount of requests, one third of the requests have failed. Error code `KPositionQualityLoss` will be returned.

It is also possible not to simulate errors. This is specified by setting `Deterministic=0`.

Horizontal and vertical accuracy, are same for all location estimates. For each new location estimate, horizontal accuracy is multiplied with a random generated value. The multiplied value is added as “error” for longitude and latitude to the location.

Altitude always starts at 0 meters over the WGS-84 ellipsoid model. For each new location estimate, vertical accuracy is multiplied with a random generated value. The multiplied value is added to the current altitude.

If the syntax of the file is incorrect or all settings are not present, the system wide error code `KErrCorrupt` is returned. This error code is returned when opening the positioner, `RPositioner::Open()`.

For example, if you want to simulate the locations of a car that drives 20 m/s (72 km/h) straight to the North and every seventh location request fails:

```
Horizontal accuracy=20;
Vertical accuracy=30;
TimeToFix min=2;
TimeToFix max=7;
Powerup time=3;
Longitude=11.34;
Latitude=57.11;
Speed=20;
Course=0;
Deterministic=7;
```

Using this example would give locations between 2 and 7 seconds at each location update that is requested, except for the first request when a powerup time of three seconds is added.

3.6.3 Simulation PSY Route plug-in user's guide

3.6.3.1 Introduction

The Simulation PSY (SimPSY) Route plug-in enables you to define routes and movement directions which can be passed transparently to the emulator's SimPSY plug-in for positioning simulation.

The Simulation PSY (=Positioning Plugin) provides a user interface to simulated location data. This section contains instructions on how to use the location interface.

3.6.3.2 SimPSY Route dialog

The SimPSY Route Dialog provides tools for developing location-based applications with the SimPSY plug-in. To access the Route dialog, select Tools > Utilities from the emulator main menu, and click the Route tab. The following dialog is displayed:

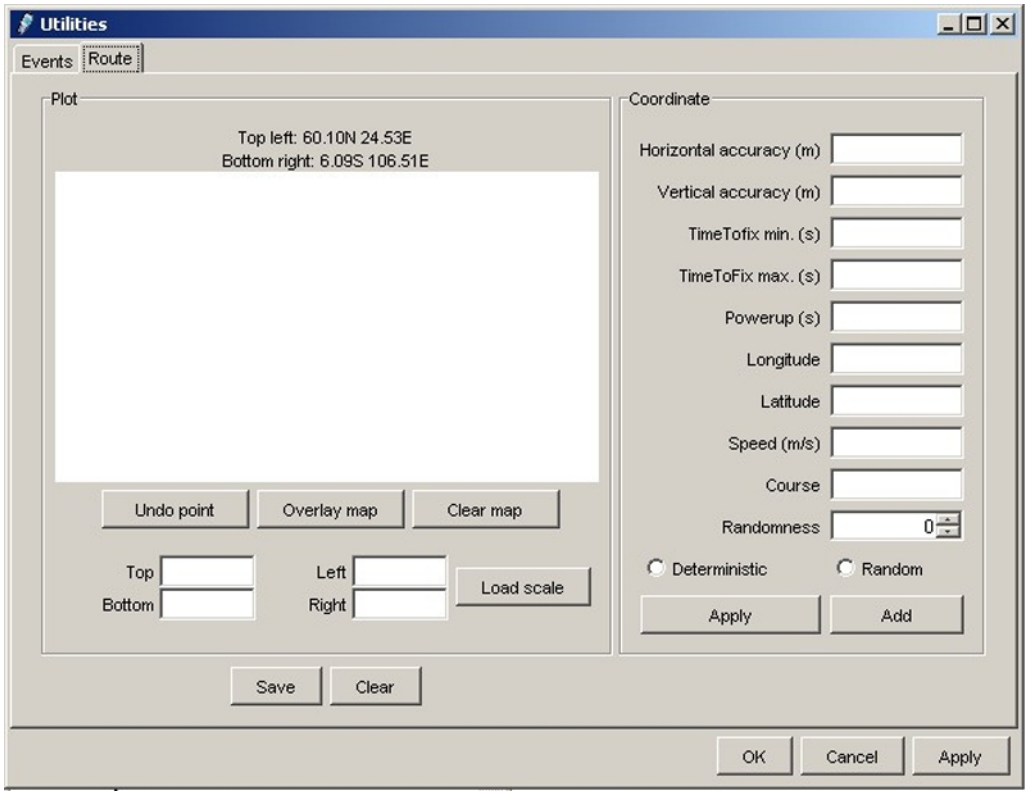


Figure 107: SimPSY Route dialog

The plug-in outputs in NMEA or SPS file format (see the Simulation PSY user's guide for a more detailed description of the formats). The file formats support somewhat different data: NMEA specifies single coordinates from a GPS device, while SPS is an artificial format that specifies data, such as speed and direction. Consequently, NMEA is recommended as export format for complex routes.

3.6.3.3 Defining location information using the Plot field

The **Plot** field on the right side of the Route dialog includes an area, which is used for drawing the route, fields for entering the corner coordinates of the area, and buttons for route-related functions.

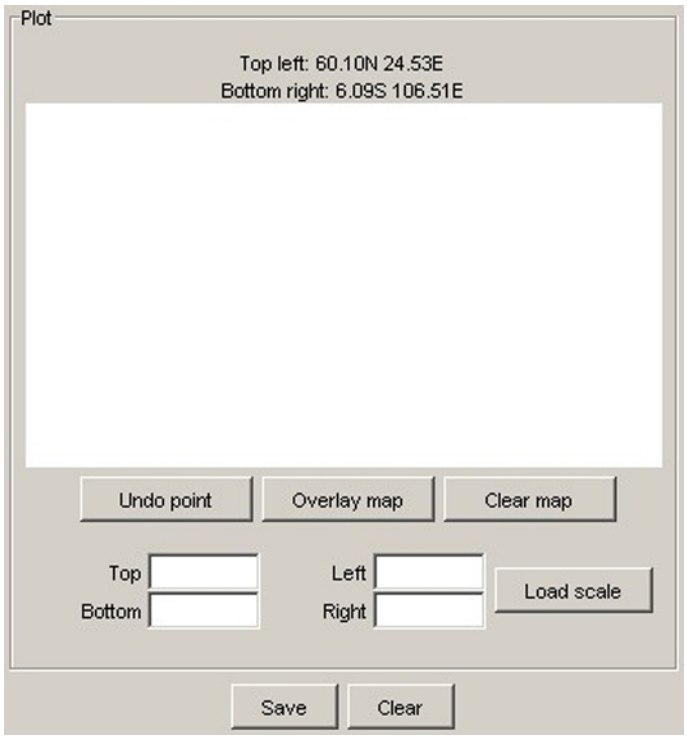


Figure 108: Plot field in SimPSY Route dialog

The current coordinates are displayed above the drawing area. To change the coordinates, enter new coordinates, either using the “N”, “S” notation or by using negative and positive values in the fields, and click **Load scale**. The new coordinates are displayed above the drawing area. Note that loading a new scale deletes all existing coordinates in the route, as scaling them might easily cause confusion.

The **Undo point** button undoes the last coordinate and is unlimited, that is, you can undo settings all the way to the first coordinate. The **Clear** button clears all existing coordinates from the route.

To get a more realistic view of the route, you can overlay an image on the drawing. To overlay an image, click the **Overlay map** button and then select the image to overlay. The supported file formats are `JPEG`, `PNG` and `GIF`. It will work even though the image is not of the right size (the size of the window is typically 377x200 pixels.) Overlaying the image does not affect the output of the plug-in.

3.6.3.4 Defining location information using the Coordinate field

The **Coordinate** field on the right side of the Route dialog dialog displays all the parameters for the most recently entered coordinate for the route.

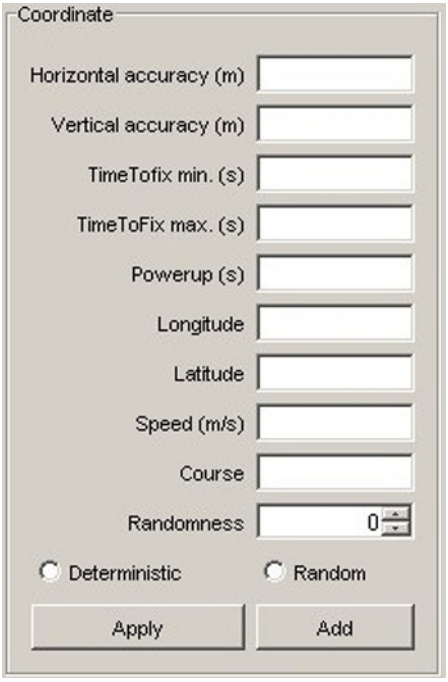


Figure 109: Plot field in SimPSY Route dialog

To add a new coordinate for the route, enter the required values in the fields, and click **Add**. To modify the parameters of the most recently added coordinate, enter the required values, and click **Apply**. Note that as the drawing of the route alone cannot present all the different options available, default values will be substituted in the added coordinates.

The values of the most recently added coordinate function as the default values for the next coordinate to be added. For example, if you add a coordinate by drawing it, change the speed slightly, and click **Apply**, the new speed will be the default speed for the subsequently added coordinates. Note that the direction changes dynamically when you add new coordinates, in other words, the direction value of the old coordinate is in the direction of the new coordinate.

For more detailed information on the different parameters, see the Simulation PSY user's guide. Some parameters are only valid for the SPS file format and are subsequently ignored when writing output in NMEA format.

3.6.3.5 Saving the location file

When you are finished with defining the route, save it in the desired format (for example, click **Save** and enter a filename with the suffix `.nme` file extension) and then, click the **Apply** button in the SimPSY Route dialog .

Clicking **Apply** communicates the updates for the emulator and configures SimPSY to use the newly created file for location information. Thus, you do not have to manually instruct SimPSY to load the new location file, you can directly start fetching the coordinates. When saving the location file it is important that you use the `.nme` file extension for NMEA files and the `.sps` file extension for SPS files, as the SimPSY Configurator displays files of these types only, and thus does the route plug-in.

3.6.3.6 Testing the location information in the emulator

To test the location data that you have created, first, make sure that you have the "Simulation PSY" option set as the location positioning method in the emulator by going to Configurations in the main display of the emulator and selecting **Location > Settings > Positioning method**. Then, open the SimPSYConfigurator in the emulator and select **Get current location**. A coordinate value should then be returned.

3.6.3.7 Using SimPSY Route with several IDEs

The emulator's c: drive is detected based on the current path and hard coded search paths. This does not normally constitute a problem, but if you wish to use both CodeWarrior and Visual C++ on the same PC, you must manually check that the path is correct when saving the location file (`winscw/c` for CodeWarrior, `wins/c` for Visual C++.)

3.6.4 Simulation PSY Configurator user's guide

3.6.4.1 Introduction

This document describes the usage of the Simulation PSY Configurator application. It is valid from S60 release 3.1 and onwards.

A developer of a location based application needs to retrieve positions during the development stage. With Simulation PSY it is possible to get positions when running applications in the Windows emulator environment and also on the hardware.

When testing, Simulation PSY enables simulation of different positions without actually needing to go to these different locations. With this software, the retrieval of positioning data is independent of any hardware.

Simulation PSY makes it easy to reproduce different locations, for example, to simulate positions on the North Pole, or to test how an application behaves when moving over the Zero Meridian. Without Simulation PSY it is hard to test applications in this kind of situations. It is useful to test how a location application behaves when changing the accuracy of the location estimates. This is possible with Simulation PSY. It is also possible to test error situations.

The Simulation PSY Configurator application has been implemented for configuring Simulation PSY. It allows for the selection of the data file for Simulation PSY. Data formats of are described in S60 Simulation PSY User's Guide. The application cannot be seen in Application Shell. It can be launched from the 'Position Method Settings' view, either through General Settings or through Location Settings Launch API, as an embedded application.

3.6.4.2 Installation

The Simulation PSY Configurator application requires that Location Framework is present in the environment. A SISX package `simpsyconfigurator.sisx` is available for hardware installation. It installs Simulation PSY and Simulation PSY Configurator. It also installs the necessary data files that are specified in Preinstalled simulation files. Location Framework is present by default, but if it is missing it should be installed before using this application.

3.6.4.3 Launching the Simulation PSY Configurator

Context

The Simulation PSY Configurator application is not visible in Application Shell and can be launched only from the Location application. The Location application launches the PSY Configuration UI application as an embedded application.

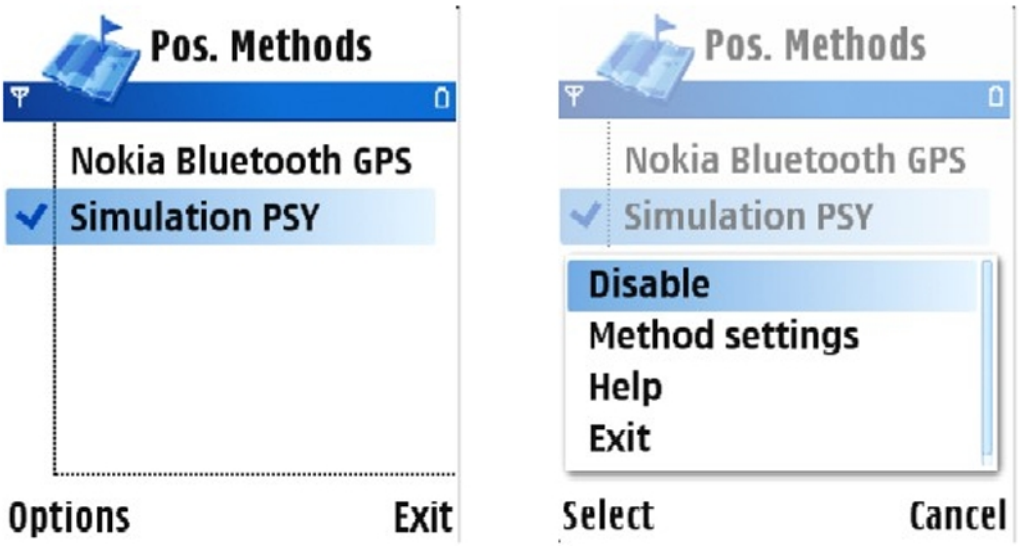


Figure 110: Launching of Simulation PSY Configurator

Launching the PSY Configuration UI application:

Steps

1 In the Location applications main view, highlight **Simulation PSY**.
.

2 Click **Options**.
.

The menu pops up.

3 Select the **Method Settings** item to open the Simulation PSY Configurator.
.

3.6.4.4 Main view

The Simulation PSY Configurator application main view is as shown below. The main view contains a listbox showing the currently configured Simulation PSY file.

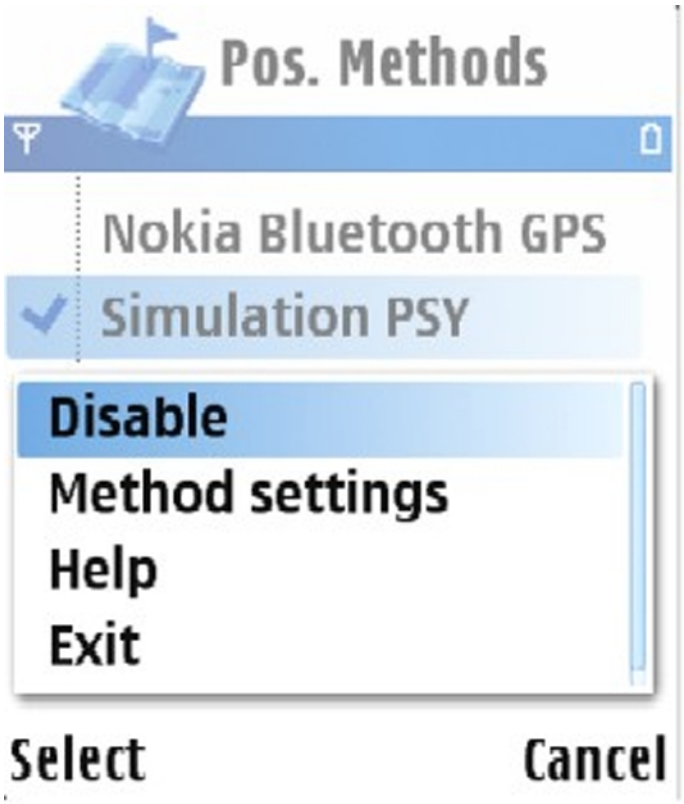


Figure 111: Simulation PSY Configurator main view

3.6.4.5 Simulation PSY configuration

The Simulation PSY **Configuration** dialog consists of two **Options** menu items, as described in the following sub-sections.

Select Configuration File

This enables the user to select the data file to be used when giving the simulated location information.

Configuring the data file:

1 Select this command from the **Options** menu.

A **Select file** dialog pops up.

2 Choose one file and press **Select**.

The value will be shown in the edit box in the form.

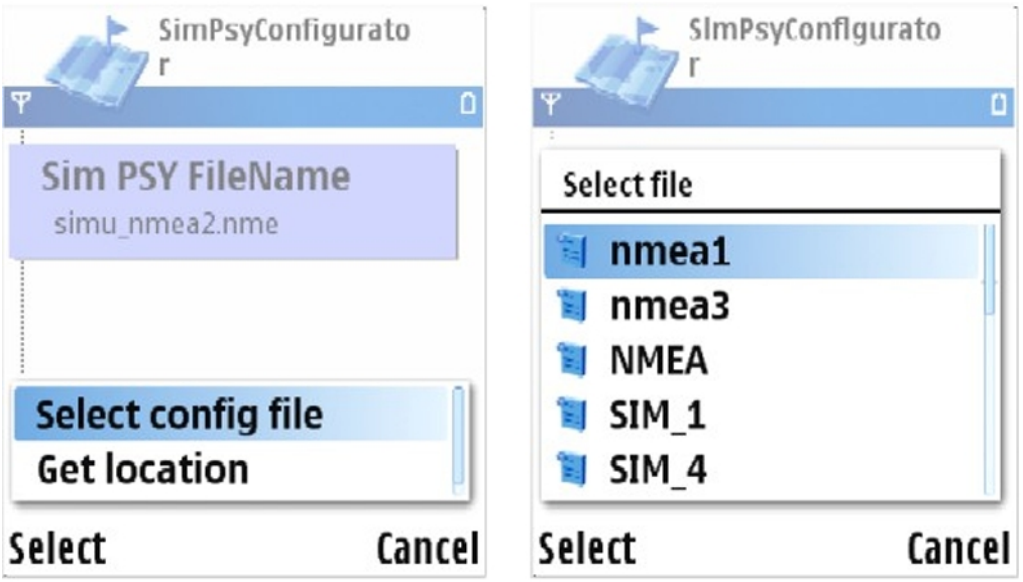


Figure 112: Select Configuration file

Get Current Location

This option gives the simulated location using the selected data file. This gives the user a way of verifying that the correct data file is selected.

Select the **Get Location** menu item from the **Options** menu in the **Configuration** dialog. If the location information is available the simulated latitude, longitude, altitude, horizontal accuracy and vertical accuracy are displayed in a list dialog in the specified order.

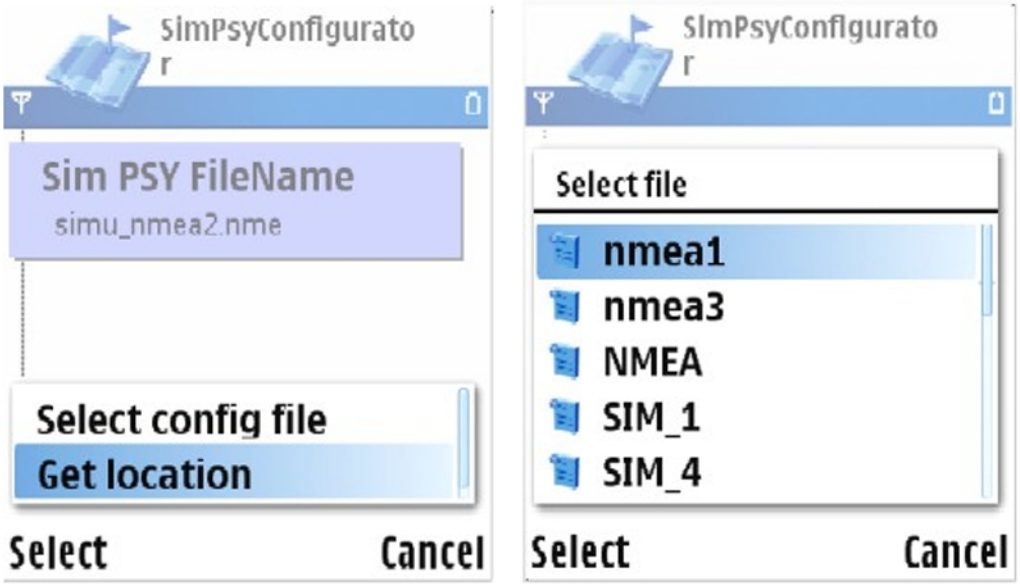


Figure 113: Get current location

In case of error, an error note with the error code will be displayed.

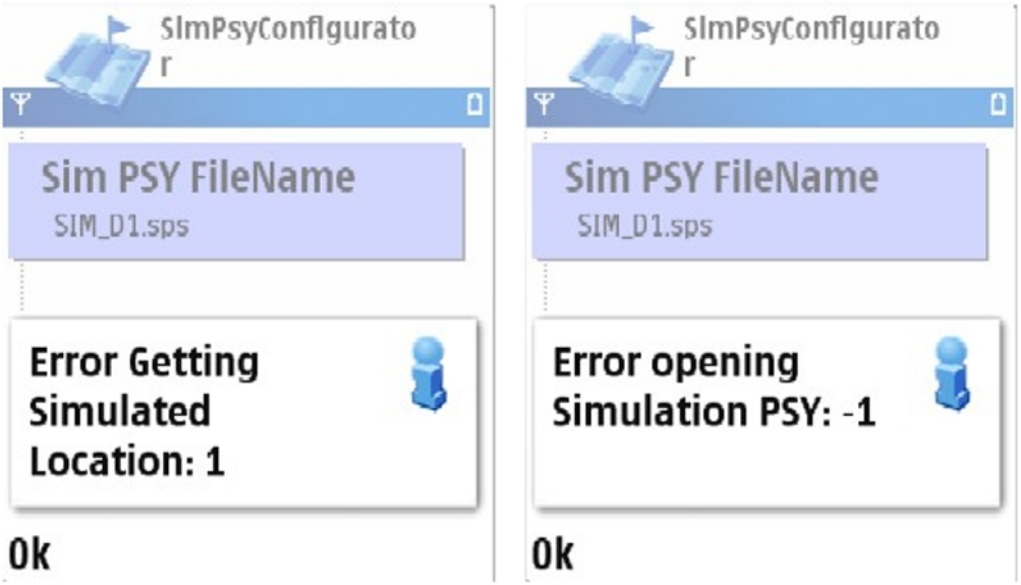


Figure 114: Error note

Simulation PSY Configurator always uses Simulation PSY to get the location information. Therefore the values are simulated values and not the actual location.

	Note: After each location request has been completed Simulation PSY session is closed. Every time the user makes a request, Simulation PSY starts the simulation from the beginning. The end user will get the same location information even if a simulated movement file is used, unless the data file specifies random behavior.

3.6.4.6 Preinstalled simulation files

Simulation PSY Configurator consists of certain preinstalled simulation data files that can be used by Application Developer.

NMEA files

Simulation PSY Configurator comes preinstalled with the following NMEA files that provide valid location information.

NMEA File Name	NMEA File Name
`NMEA.nme`	All NMEA sentences in this file are valid sentences with Latitude, Longitude, movement and speed information.

SPS files

Simulation PSY Configurator consists of the following simulation files.

NMEA File Name	NMEA File Name
`simu_move_const.sps`	The location starts at Latitude=61.5285 and Longitude=23.9385 and moves with a bearing of 113.9. The speed is set to zero.
`simu_move_speed.sps`	The location starts at Latitude=78.3385 and Longitude=56.3385 and travels in the same direction at a speed of 100.

3.6.4.7 Adding Simulation PSY data files manually

The Simulation PSY Configurator installation has a set of pre-configured data files that are specified in Preinstalled simulation files. The user can also create data files (NME or SPS) that can be added by copying the files to the `c:\system\data` directory. The files must have an extension of either NME or SPS; otherwise Simulation PSY Configurator does not find them. The data file format is described in S60 Simulation PSY User's Guide.

3.7 Tools for SIP Emulation

3.7.1 Introduction

The aim of this SIP Developer User's Guide is to guide you through your first steps in setting up an environment for SIP application development for the S60 platform with the SDK. This User's Guide provides:

- The settings necessary to configure a SIP development environment.
- Information on administrative issues such as starting and stopping the SIP Server Emulator, configuration and log reading.
- Information on using the SIP Server emulator's graphical user interface (GUI).

Note: The Example Application package provides the SIP Example, which can be used for testing SIP functionality for S60. The example application itself is not described in this document: Please refer to the documentation in the Example Application package for details regarding the example application.

The extracts of code in this document are mainly from C++. For a Java user, this guide is mainly useful as regards information on SIP Server configuration.

What You Should Already Know

You should have some knowledge about C++ programming for the S60 platform and/or Symbian OS. For an introduction on programming on the S60 platform, please refer to the S60 3rd Edition SDK for Symbian OS Getting Started Guide. Familiarity with SIP architecture and SIP APIs is also recommended.

What You Will Need

Installation requirements for running the SIP example successfully are listed below. For detailed information on SDK installation, please refer to the S60 3rd Edition SDK for Symbian OS Installation Guide. The feature currently requires multiple PCs for testing. You will need one PC for each emulator instance running a SIP application; in addition you will need a separate PC running the SIP Server Emulator. For the most common use case (2 SIP clients are exchanging messages) you will need 3 separate PCs.

In addition, one should have basic knowledge on how to configure network settings in the emulator environment (see the *Emulator Connectivity* section in the *Emulator Guide*).

3.7.2 Overview of SIP for S60 3rd Edition

Session Initiation Protocol (SIP) is a protocol that initiates sessions between terminals. The initiation of sessions is established through the SIP Server Emulator. Once a session has been initiated, terminals may continue to interact without the SIP Server Emulator.

The SIP Server Emulator combines the functionality of a SIP Proxy Server and a SIP Registrar Server. It is the forwarding center that delivers SIP messages to its recipients. Incoming and outgoing SIP messages can be logged into a log file. The SIP Server Emulator can also be used in automatic response mode. When the SIP Server Emulator

receives a SIP message to be forwarded, it responds with a specified SIP message and does not forward the incoming SIP message. This automatic response mode is used mainly for testing SIP error messages.

When combined, the SIP Server Emulator and S60 SDK Emulator form an environment in which you can test your SIP applications. The SIP Example (provided in the Example Application package installed by default with the SDK) can be used for testing SIP functionality for S60. The example application itself is not described in this document: Please refer to the documentation in the Example Application package for details regarding the example application.

3.7.3 Installation and configuration

3.7.3.1 SIP developer environment setup

This section describes how to set up a SIP Development environment for the S60 3rd Edition SDK, Supporting Feature Pack 1. Building the entire environment consists of a series of actions that occur in a specific order. The steps include installing all the required software components and configuring them so that they work together.

As a result of the actions described in this section, the developer can access the environment using two PCs (see figure below).

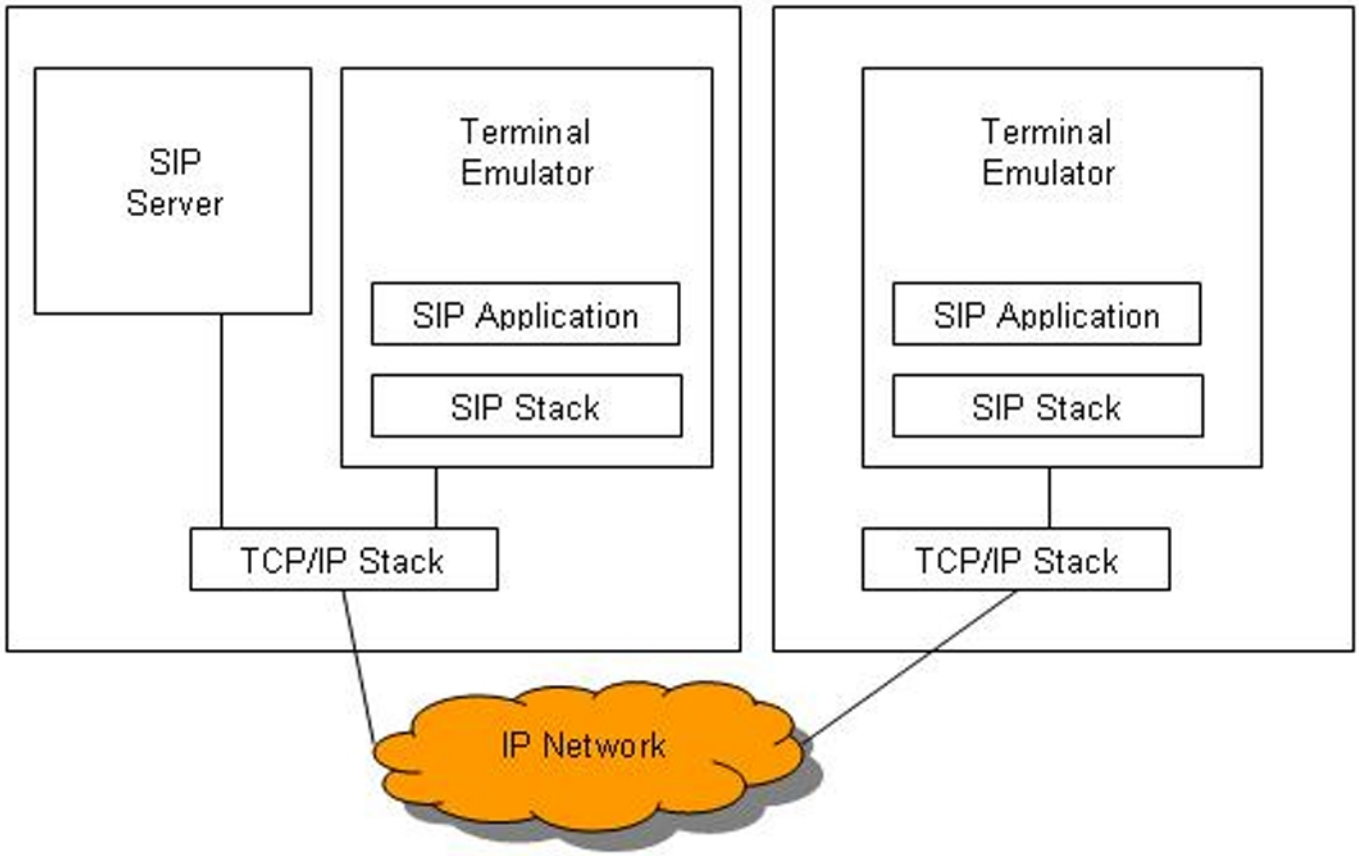


Figure 115: S60 SIP development environment

The development environment on the developer's PC will have:

- A SIP Server (that is, a SIP Server Emulator) with SIP message automatic response feature.
- Two S60 SDK emulators (running on separate PCs).
- SIP APIs that enable SIP features in S60 applications.
- An example SIP application, which makes it possible to check that the environment is correctly configured.

- Correctly configured network settings and SIP settings. This enables network communication between the Terminal Emulators and the server.

The 4-step setup procedure is introduced in the Installation and configuration section

3.7.3.2 Installation and configuration

Context

The 4-step setup procedure is introduced in this section. The setup procedure consists of the following steps:

Steps

1 Install S60 SDK.

. If not installed already, install the S60 SDK.

2 Build the example SIP applications.

. Example applications can be built by going to the main groupdirectories under the root directories of the example applications and run the following commands in the command prompt:

- `bldmake bldfiles`
- `abld build winscw udeb`

For detailed information about the example applications, please refer to the documentation provided in the Example Application package.

3 Start the SIP server emulator.

. Start the SIP Server Emulator (GUI) from the **Start** menu. Before starting the server, you must configure the domains to be used. Find out the IP address of each PC and add these to the list of domains in the SIP Server Emulator GUI (see the *SIP Server Emulator user interface* figure and SIP Server Emulator for more detailed information). Removing the default domains (domain1 and domain2) is not required.

Start the SIP Server Emulator by clicking the **Start server** button on the toolbar. The SIP Server Emulator should now accept SIP requests.

For more information on how to use the SIP Server Emulator see SIP Server Emulator.

4 Configure the SIP settings and test the environment.

. On both emulators, set the SIP configuration through **Applications > Configurations > Settings > Connection > SIP settings** if the SIP Server is located on a PC, server address is the same as the PC's IP address:

```
Profile name: Provider

Service profile: IETF

Default access point: Winsock

Public user name: player1@<server_address>

Use compression: No

Registration mode: When needed

Security negotiation: Inactive

Proxy server:

       Proxy address:<server_address>

       Realm:<server_address>

       Username: None
```

```
        Password: None

        Loose routing: Yes

        Transport protocol: TCP

        Port: 5060

Registrar server:

        Registrar address:<server_address>

        Realm:<server_address>

        Username: None

        Password: None

        Transport protocol: TCP

        Port: 5060
```

On Emulator B, the name "player1" should be replaced with another name, such as "player2". "realm" can be replaced with any text. Player1 and player2 can have different realms.

Now you can start to test the configuration with the SIP example application. The example and its documentation are provided as part of the Example Application package.

3.7.4 Using the SIP Server Emulator

3.7.4.1 SIP Server Emulator

The SIP Server Emulator UI can be started from the **Start** menu of the system: **Start > Programs > S60 Developer Tools > 3rd Edition SDK > MIDP > Tools > Sip Server**.

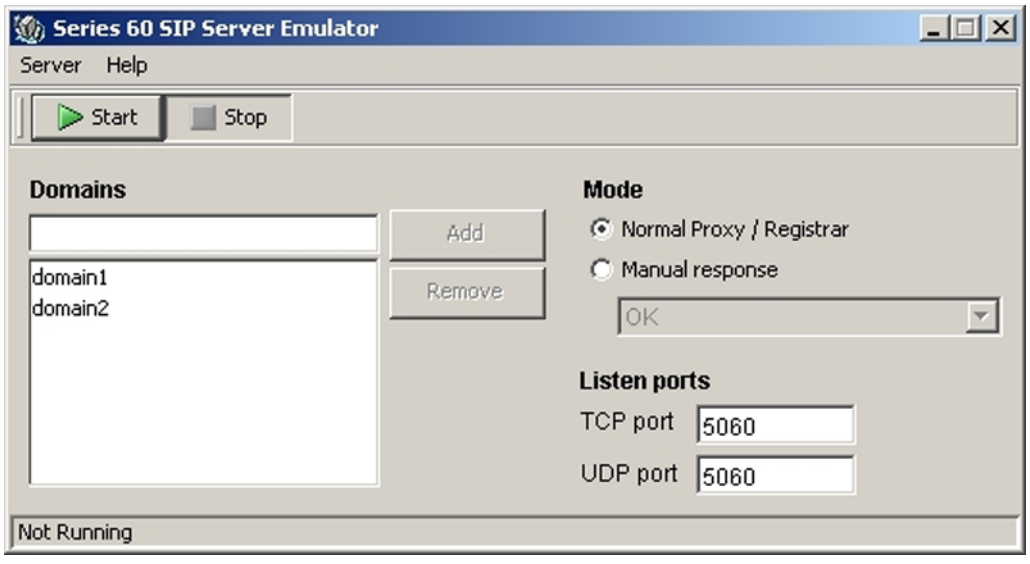


Figure 116: SIP Server Emulator user interface

3.7.4.2 Configuring the SIP Server Emulator

SIP Server Emulator must be configured before it can properly serve SIP messages.

Domains

The SIP Server Emulator acts as a Proxy/Registrar server at a selected list of domains or IP addresses. When the SIP Server Emulator UI is started, two default domains are listed.

	**Note:** The IP address of the PC running the SIP Server Emulator must be added to this list!

New domains can also be added while the SIP Server Emulator is running. An added domain will become active when the **Add** button is clicked. If you want to remove domains from the list, the SIP Server Emulator has to be stopped first. To remove a domain name, select the domain name to be removed and click the **Remove** button. Domain names cannot contain empty spaces; empty spaces are removed automatically before starting the SIP Server Emulator.

Listen Ports

The SIP Server Emulator listens to SIP messages in selected TCP and UDP ports. The default port for both protocols is 5060; port numbers have to be set before the SIP Server Emulator is started.

Ports must be numerical and cannot contain empty spaces. Empty spaces are removed automatically before starting up the SIP Server Emulator. Port configuration can be modified only when the SIP Server Emulator is not running. Port configuration becomes active when the SIP Server Emulator is started.

Mode

SIP Server emulator can act in two modes:

- In **Normal Proxy / Registrar** mode the SIP Server Emulator acts like an ordinary SIP Registrar and SIP Proxy servers would. It registers SIP addresses and forwards SIP messages.
- In the **Manual response** mode the SIP Server Emulator responds to incoming SIP messages with selected responses instead of automatically forwarding the messages. These manual responses can be selected from the **Manual response** combo box. A set of typical SIP response scenarios is available. Each SIP response scenario contains message chains with one or more SIP messages; a simple example of an error scenario could contain the messages “100 Trying” and “404 Not Found”. Some scenarios contain messages that wait for input from the Terminal Emulator before sending the rest of the messages in the message chain. If a response from the combo box is changed when a scenario is not finished, the SIP Server Emulator must be restarted before the new response configuration becomes active

3.7.4.3 Starting the SIP Server Emulator

Context

	**Note:**

The SIP Server Emulator cannot be started if it is already running.

Steps

1 Select **Server > Start server** .

.

Alternative steps

- Click the **Start** button.

3.7.4.4 Stopping the SIP Server Emulator

Context

	**Note:**

Once you stop the emulator, all outgoing traffic stops immediately and no new SIP messages are forwarded.

Steps

1 Select **Server > Stop server** .
.

Alternative steps

- Click the **Stop** button.

Alternative steps

- You can also shut down the SIP Server Emulator by selecting **Server > Exit** or closing the SIP Server Emulator UI window.

4 Java™ Technology for S60: Basic concepts

4.1 Mobile Information Device Profile (MIDP)

The Mobile Information Device Profile (MIDP) is a set of Java APIs that together with the Connected Limited Device Configuration (CLDC) provides a complete Java™ ME application runtime environment for mobile information devices (MIDs). The MIDP specifications (JSR-000118) define minimum hardware, software, networking, application requirements, as well as standard system APIs for devices that support MIDP.

MIDP is intended for 'connected' devices such as smartphones, which are characterized by having limited Central Processing Unit (CPU), memory, keyboard, and display capabilities.

For more details on MIDP and the related specifications, please refer to the Sun Developer Network at http://java.sun.com/products/midp/ .

4.2 Connected Limited Device Configuration (CLDC)

Connected Limited Device Configuration (CLDC) defines the Java language features and the core libraries of the Java Virtual Machine (JVM).

4.3 MIDlet

Java applications developed with the S60 SDK are called MIDlets. A MIDlet is an application, which has been written for the Java™ Micro Edition Mobile Information Device Profile (MIDP).

A simple example of a MIDlet is the HelloWorld example application provided with the S60 SDK.

4.4 MIDlet lifecycle

MIDlets are designed to co-exist with other applications on a Mobile Information Device (MID). The MIDlet lifecycle is pivotal in enabling this co-existence. Basically, a MIDlet can be in one of the following states: paused, active, or destroyed.

The Application Management Software (AMS) controls the state of a MIDlet by directing a MIDlet to start, pause, or destroy itself. It constructs a MIDlet and communicates changes in its state by calling the following methods, which every MIDlet must implement: `startApp()`, `pauseApp()`, and `destroyApp()`.

All MIDlets start out in the paused state. When a user starts an application, the AMS calls the `startApp()` Method of the MIDlet in question, after which the MIDlet is in active state (that is, the application is opened).

startApp() method

When a user opens a Java application on her device, the AMS calls the `startApp()` method. In this case, the MIDlet is newly constructed with the method. In the case of such a newly constructed MIDlet, the `startApp()` method calls `Display.getDisplay(this).setCurrent(startup_screen)` to invoke the startup screen of the MIDlet (that is, application) in question.

The `startApp()` method is also invoked by the AMS when a MIDlet is returned from the paused state. In this case, the `startApp()` method will typically need to restore the state it was in before it was paused.

pauseApp() method

When a user pauses (but does not close) an application, the AMS calls the `pauseApp()` method. In this case, the application is paused and it is not longer displayed in the User Interface (UI) to the user. However, any threads or timers created by the MIDlet in question will continue to run unless the user stops these manually. These threads keep on running by default, because the application will need to restore its state when it is reactivated with the `startApp()` method.

destroyApp() method

When a user closes an application, the AMS calls the `destroyApp()` method. This method instructs the MIDlet to release all the resources it has reserved and to close down as soon as possible. In effect, this means that the application in question closes all input and output streams, terminates all threads, and cancels all timers. Once the `destroyApp()` method has been called, the MIDlet ceases to access the display, in other words, the application is no longer displayed in the UI to the user.

4.5 MIDP user interface APIs

The MIDP User Interface (UI) has a high-level and low-level API.

The high-level API classes `Alert`, `Form`, `List`, and `TextBox` are extensions of the abstract class `Screen`. These are designed to provide abstractions and components that are highly portable across different Mobile Information Devices (MIDs), as the application takes care of aspects such as drawing, font characteristics, navigation, and scrolling. A particular device's implementation of these classes performs the adaptation to its hardware and native UI look and feel.

The low-level API is based on use of the abstract class `Canvas`. In comparison with the high-level APIs, this class allows applications to have more direct control of the UI by permitting greater control of what is drawn on the display and reception of low-level keyboard events.

4.6 MIDlet suites and application descriptors

A MIDlet suite is a Java archive (JAR file), which contains one or more MIDlets. All MIDlets are deployed to devices as suites. MIDlets in the same suite share the same execution environment (virtual machine) and can interact with one another.

Each MIDlet suite JAR file has an associated Application Descriptor (JAD file) used to describe its contents. The application descriptor MIME type is `text/vnd.sun.j2me.app-descriptor`. The file extension of such an application descriptor must be `.jad`. The AMS uses the descriptor to manage each MIDlet included in the suite, that is, to verify that a MIDlet is suited for execution on the device before loading the MIDlet suite JAR file.

4.7 HelloWorld MIDlet

The HelloWorld application is a simple MIDlet included with the SDK as an example application. When opened, the application displays a "Hello World!" message on the display of the device. The application is closed by selecting the **Exit** command of the right softkey.

The structure of the HelloWorld application is depicted below:

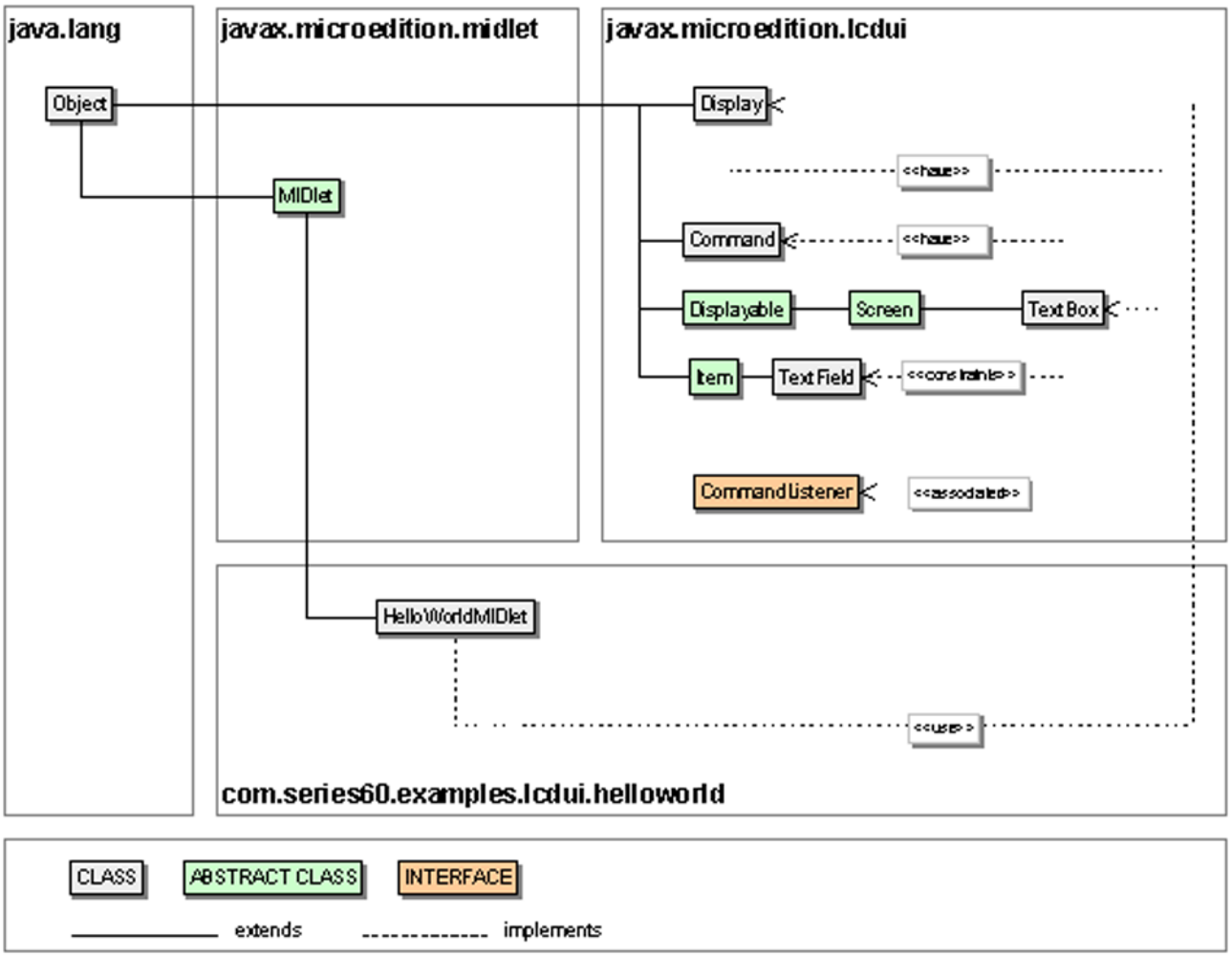


Figure 117: Structure of the HelloWorld MIDlet

The `HelloWorldMIDlet` class extends the `javax.microedition.midlet.MIDlet` class. It uses the MIDP high-level API `javax.microedition.lcdui.TextBox` class to enter and edit text, in this case, "Hello World".

The `TextBox` class extends the abstract class `Screen`, which, in turn, extends the abstract class `Displayable`. The `TextBox` UI component has a string contents area that allows users to enter and edit text.

The `HelloWorldMIDlet` class also implements the `CommandListener` interface used to receive user input from the command area.

The functionality of the `HelloWorldMIDlet` is the following: The `startApp()` method sets the MIDlet's current `Display` to the `HelloWorldMIDlet` object, if a current display does not already exist. If the `HelloWorldMIDlet` object receives an Exit command from the user interface, it calls the `commandAction()` method of the `HelloWorldMIDlet`. The `HelloWorldMIDlet` then handles the MIDlet state transition to a Destroyed state. See below for the `HelloWorldMIDlet.java` example code.

```
/* Copyright (c) 2003, Nokia. All Rights Reserved */
package com.series60.examples.lcdui.helloworld;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 * This class illustrates the implementation of a simple MIDlet that
 * displays a "Hello World" message to the screen
 * This class extends the class javax.microedition.midlet.MIDlet. It
 * creates a TextBox object and displays it on screen.
```

```
 * <p>
 * Additionally this MIDlet provides a way for the user to exit the
 * MIDlet
 *
 * @author Nokia
 * @version 1.0
 */
public class HelloWorldMIDlet extends MIDlet implements CommandListener {

    /** create a hello world string */
    public final static String HELLO_STRING = "Hello World";

    /** create a default value for the priority of a command */
    private final static int DEFAULT_COMMAND_PRIORITY = 1;

    /** the exit button command */
    private final static Command EXIT_COMMAND = new Command("Exit",
Command.EXIT, DEFAULT_COMMAND_PRIORITY);

    /**
     * Creates the new text box, then set it to
     * current active component on screen
     */
    public void startApp() {
        /*
         *  Create a textbox
         *  titled HelloWorldMIDlet
         *  the initial value "Hello World"
         *  max input the length of "Hello World"
         *  and uneditable
         */
        TextBox textBox = new TextBox("HelloWorldMIDlet", HELLO_STRING,
HELLO_STRING.length(), TextField.UNEDITABLE);

        /** Add the exit command to the TextBox */
        textBox.addCommand(EXIT_COMMAND);

        /** Set this instance to be the event handler */
        textBox.setCommandListener(this);

        /** Set the display to show the textbox */
        Display.getDisplay(this).setCurrent(textBox);
    }

    /**
     * Performs no operation, because there are no background
     * activities or record stores to be closed.
     */
    public void pauseApp() {
    }

   /**
     * Performs no operation because there is nothing which is not handled
     * by the garbage collector.
     *
     * @param unconditional ignored.
     */
    public void destroyApp(boolean unconditional) {
    }

    /**
     * This is performed when a button is pressed
     *
     * @param cmd the command
     * @param source ignored
     */
    public void commandAction(Command cmd, Displayable source) {
        if (cmd == EXIT_COMMAND) {
            destroyApp(false);
            notifyDestroyed();
        }
    }
}
```

5 Developer Resources

5.1 Sun Java Developer website

The Java Developer Home website at http://developers.sun.com/index.html provides an abundance of information and support on Java technology. The Mobility Development Center at http://developers.sun.com/techtopics/mobility/ at this site is a definitive resource for developers seeking to create, test, certify, and market Java mobility applications.

5.2 S60 website

The S60 website at http://www.series60.com provides a wealth of information for developers designing applications for S60 smartphones: device specifications, featured tools and SDKs, documentation targeted for developers and application testing instructions. You will also find information related to S60 terminal manufacturers, operators, enterprises, product creation community and detailed information on the different S60 devices.

5.3 Symbian Developer network

The Symbian Developer network at http://developer.symbian.com/main/getstarted/ provides downloads, technical papers, system documentation as well as more interactive elements such as newsletters and discussion forums.

- Tools and SDKs at http://developer.symbian.com/main/tools/index.jsp - Developer tools, utility libraries and API extensions, example applications, and open source projects
- Library at http://developer.symbian.com/main/oslibrary/index.jsp - Symbian OS documentation and technical papers.
- Academy at http://developer.symbian.com/main/academy/index.jsp - Symbian Press, Symbian Academy, accredited Symbian developer, FAQs, training courses.
- Symbian Signed at http://developer.symbian.com/main/signed/index.jsp - Symbian Signed is an industry-backed program that allows you to certify your Symbian OS applications so that they can be installed and used on S60 3rd Edition and UIQ 3 phones. Certifying an application via Symbian Signed can greatly improve the application's route and time to market.
- Forums at http://developer.symbian.com/forum/index.jspa - Discussion groups and feedback areas.
- Symbian Developer Network Wiki at http://developer.symbian.com/wiki/dashboard.action

5.4 Books on Java

- *Professional Java Mobile Programming*

  by Ronald Ashri, Steve Atkinson, Rob Machin, Martin Graf, Marten Haglind, Nadia Nashi, Richard Taylor, Danny Ayers, Bill Ray, Chanoch Wiggers

  Wrox Press Inc; 1st edition (July 2001).
- *Learning Wireless Java*

  by Qusay Mahmoud

  O'Reilly; 1st edition (December 15, 2001)
- *Professional Mobile JAVA with J2ME*

  by Mikko Kontio

  Cromland; Book and CD Rom edition (October 15, 2002)
- *J2ME in a Nutshell (O'Reilly Java)*

  by Kim Topley

  O'Reilly; 1st edition (March 1, 2002)

- *Wireless Java for Symbian Devices*

  by Jonathan Allin, Colin Turfus, Alan Robinson, Lucy Sweet, John Brown

  John Wiley & Sons; 1st edition (September 20, 2001)
- *Enterprise J2ME: Developing Mobile Java Applications*

  by Michael Juntao Yuan

  Prentice Hall PTR; 1st edition (October 20, 2003)
- *Sams Teach Yourself Wireless Java with J2ME in 21 Days*

  by Michael Morrison

  Sams; Book and CD Rom edition (June 27, 2001)
- *MIDP 2.0 Style Guide for the Java 2 Platform, Micro Edition*

  by Cynthia Bloch, Annette Wagner

  Addison-Wesley Professional; 1st edition (June 10, 2003).

6 Legal Notes

6.1 Nokia Corporation End-User Software Agreement

This Software Agreement ("Agreement") is between You (either an individual or an entity), the End User, and Nokia Corporation ("Nokia"). The Agreement authorizes You to use the Software specified in Clause 1 below, which may be stored on a CD-ROM, sent to You by electronic mail, or downloaded from Nokia's Web pages or Servers or from other sources under the terms and conditions set forth below.

This is an agreement on end-user rights and not an agreement for sale. Nokia continues to own the copy of the Software and the physical media contained in the sales package and any other copy that You are authorized to make pursuant to this Agreement. You also acknowledge that the Software contains portions of software licensed by Symbian Software Limited whose registered office is situated at 2-6 Boundary Row, London SE1 8HP, UK ("Symbian") or its third party licensors and which has been licensed to Nokia under a separate agreement between Nokia and Symbian.

Read this Agreement carefully before installing, downloading, or using the Software. By clicking on the "I Accept" button while installing, downloading, and/or using the Software, You agree to the terms and conditions of this Agreement. If You do not agree to all of the terms and conditions of this Agreement, promptly click the "Decline" or "I Do Not Accept" button, cancel the installation or downloading, or destroy or return the Software and accompanying documentation to Nokia. YOU AGREE THAT YOUR USE OF THE SOFTWARE ACKNOWLEDGES THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS.

1 **SOFTWARE**

As used in this Agreement, the term "Software" means, collectively: (i) the software product identified above (ii) all the contents of the disk(s), CD-ROM(s), electronic mail and its file attachments, or other media with which this Agreement is provided, including the object code form of the software delivered via a CD-ROM, electronic mail, or Web page (iii) digital images, stock photographs, clip art, or other artistic works ("Stock Files") (iv) related explanatory written materials and any other possible documentation related thereto ("Documentation"); (v) fonts, and (vi) upgrades, modified versions, updates, additions, and copies of the Software (collectively "Updates"), if any, licensed to You by Nokia under this Agreement.

2 **END USER RIGHTS AND USE**

Nokia grants to You non-exclusive, non-transferable end-user rights to install the Software on the local hard disk(s) or other permanent storage media of one computer and use the Software on a single computer or terminal at a time.

You acknowledge that the use of the Software may require licenses to third party patents and that Nokia does not grant any licenses to such third party patents under this Agreement to the You. Also, any and all standards related licenses with respect to Essential Patents (as defined below) are specifically excluded from the scope of this Agreement, and those licenses need to be acquired separately from Nokia or the respective right holders, as the case may be. “Essential Patent” means any patent claiming a feature necessarily and unavoidably required for compliance with industry standards (usually GSM, WCDMA or other similar mobile communications standard), to the limited extent only that infringement or use of such claims of a patent cannot be avoided in remaining compliant with industry standard either for technological reasons or for lack of commercially viable technical alternatives.

No patent licenses to any patents of Nokia and/ or its Affiliates are granted under this Agreement.

3 **LIMITATIONS ON END USER RIGHTS**

You may not copy, distribute, or make derivative works of the Software except as follows:

(a) You may not use, modify, translate, reproduce, or transfer the right to use the Software or copy the Software except as expressly provided in this Agreement.

(b) You may not resell, sublicense, rent, lease, or lend the Software.

(c) You may not reverse engineer, reverse compile, disassemble, or otherwise attempt to discover the source code of the Software (except to the extent that this restriction is expressly prohibited by law) or create derivative works based on the Software.

(d) Unless stated otherwise in the Documentation, You shall not display, modify, reproduce, or distribute any of the Stock Files included with the Software. In the event that the Documentation allows You to display the Stock Files, You shall not distribute the Stock Files on a stand-alone basis, i.e., in circumstances in which the Stock Files constitute the primary value of the product being distributed. You should review the "Readme" files associated with the Stock Files that You use to ascertain what rights You have with respect to such materials. Stock Files may not be used in the production of libelous, defamatory, fraudulent, infringing, lewd, obscene, or pornographic material or in any otherwise illegal manner. You may not register or claim any rights in the Stock Files or derivative works thereof.

(e) You agree that You shall only use the Software in a manner that complies with all applicable laws in the jurisdiction in which You use the Software, including, but not limited to, applicable restrictions concerning copyright and other intellectual property rights.

4 **INTELLECTUAL PROPERTY RIGHTS**

The Software and all rights, without limitation including title and intellectual property rights therein, are owned by Nokia and/or its licensors and affiliates and are protected by international treaty provisions and all other applicable national laws of the country in which it is being used. The structure, organization, and code of the Software are the valuable trade secrets and confidential information of Nokia and/or its licensors and affiliates. You must not copy the Software, except as set forth in clause 3 (Limitations On End-User Rights). Any copies which You are permitted to make pursuant to this Agreement must contain the same copyright and other proprietary notices that appear on the Software. You specifically agree and acknowledge that you shall include and maintain a copyright notice in any application developed with equal prominence to that given any other copyright notice as follows: "This Application includes software licensed from Symbian Software Ltd © 1998-200[4]". You may, with the prior consent of Symbian and subject to such restrictions as Symbian may impose, use the Symbian Logo in connection with the marketing, sale and distribution of Your Applications, including on such Applications and the Application packaging.

5 **MULTIPLE ENVIRONMENT SOFTWARE / MULTIPLE LANGUAGE SOFTWARE / DUAL MEDIA SOFTWARE / MULTIPLE COPIES / UPDATES**

If the Software supports multiple platforms or languages, if You receive the Software on multiple media, or if You otherwise receive multiple copies of the Software, the number of computers on which all versions of the Software are installed shall be one computer. You may not rent, lease, sublicense, lend, or transfer versions or copies of the Software You do not use. If the Software is an Update to a previous version of the Software, You must possess valid end-user rights to such a previous version in order to use the Update, and You may use the previous version for ninety (90) days after You receive the Update in order to assist You in the transition to the Update. After such time You no longer have a right to use the previous version, except for the sole purpose of enabling You to install the Update.

6 **COMMENCEMENT & TERMINATION**

This Agreement is effective from the first date You install the Software. You may terminate this Agreement at any time by permanently deleting, destroying, and returning, at Your own costs, the Software, all backup copies, and all related materials provided by Nokia. Your end-user rights automatically and immediately terminate without notice from Nokia if You fail to comply with any provision of this Agreement. In such an event, You must immediately delete, destroy, or return at Your own cost, the Software, all backup copies, and all related material to Nokia.

7 YOU ACKNOWLEDGE THAT THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW NEITHER NOKIA, ITS LICENSORS OR AFFILIATES, NOR THE COPYRIGHT HOLDERS MAKE ANY REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR THAT THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS, OR OTHER RIGHTS. THERE IS NO WARRANTY BY NOKIA OR BY ANY OTHER PARTY THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE. YOU ASSUME ALL RESPONSIBILITY AND RISK FOR THE SELECTION OF THE SOFTWARE TO ACHIEVE YOUR INTENDED RESULTS AND FOR THE INSTALLATION, USE, AND RESULTS OBTAINED FROM IT.

8 **NO OTHER OBLIGATIONS**

This Agreement creates no obligations on the part of Nokia other than as specifically set forth herein.

9 **LIMITATION OF LIABILITY**

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL NOKIA, ITS EMPLOYEES OR LICENSORS OR AFFILIATES BE LIABLE FOR ANY LOST PROFITS, REVENUE, SALES, DATA, OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, PROPERTY DAMAGE, PERSONAL INJURY, INTERRUPTION OF BUSINESS, LOSS OF BUSINESS INFORMATION, OR FOR ANY SPECIAL, DIRECT, INDIRECT, INCIDENTAL, ECONOMIC, COVER, PUNITIVE, SPECIAL, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND WHETHER ARISING UNDER CONTRACT, TORT, NEGLIGENCE, OR OTHER THEORY OF LIABILITY ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF NOKIA OR ITS LICENSORS OR AFFILIATES ARE ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME COUNTRIES/STATES/ JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF LIABILITY, BUT MAY ALLOW LIABILITY TO BE LIMITED, IN SUCH CASES, NOKIA, ITS EMPLOYEES OR LICENSORS OR AFFILIATES' LIABILITY SHALL BE LIMITED TO U.S. $50. Nothing contained in this Agreement shall prejudice the statutory rights of any party dealing as a consumer. Nothing contained in this Agreement limits Nokia's liability to You in the event of death or personal injury resulting from Nokia's negligence. Nokia is acting on behalf of its employees and licensors or affiliates for the purpose of disclaiming, excluding, and/or restricting obligations, warranties, and liability as provided in this clause 9, but in no other respects and for no other purpose.

10 **INDEMNITY**

You shall defend, indemnify and hold Nokia and its licensors harmless against any claims, damages, liabilities, losses, costs, suits or expenditures incurred by Nokia, its Affiliates, or licensors as a result of any infringement or alleged infringement of intellectual property rights of a third party caused by Your development or exploitation of the Software.

11 **TECHNICAL SUPPORT**

Nokia has no obligation to furnish You with technical support unless separately agreed in writing between You and Nokia.

12 **EXPORT CONTROL**

The Software, including technical data, includes cryptographic software subject to export controls under the U.S. Export Administration Regulations ("EAR") and may be subject to import or export controls in other countries. The EAR prohibits the use of the Software and technical data by a Government End User, as defined hereafter, without a license from the U.S. government. A Government End User is defined in Part 772 of the EAR as "any foreign central, regional, or local government department, agency, or other entity performing governmental functions; including governmental research institutions, governmental corporations, or their separate business units (as defined in part 772 of the EAR) which are engaged in the manufacture or distribution of items or services controlled on the

Wassenaar Munitions List, and international governmental organizations. This term does not include: utilities (telecommunications companies and Internet service providers; banks and financial institutions; transportation; broadcast or entertainment; educational organizations; civil health and medical organizations; retail or wholesale firms; and manufacturing or industrial entities not engaged in the manufacture or distribution of items or services controlled on the Wassenaar Munitions List.)" You agree to strictly comply with all applicable import and export regulations and acknowledge that You have the responsibility to obtain licenses to export, re-export, transfer, or import the Software. You further represent that You are not a Government End User as defined above, and You will not transfer the Software to any Government End User without a license.

13 **NOTICES**

All notices and return of the Software and Documentation should be delivered to:

NOKIA CORPORATION

P.O. Box 100

FIN-00045 NOKIA GROUP

FINLAND

14 **APPLICABLE LAW & GENERAL PROVISIONS**

This Agreement is governed by the laws of Finland. All disputes arising from or relating to this Agreement shall be settled by a single arbitrator appointed by the Central Chamber of Commerce of Finland. The arbitration procedure shall take place in Helsinki, Finland in the English language. If any part of this Agreement is found void and unenforceable, it will not affect the validity of the balance of the Agreement, which shall remain valid and enforceable according to its terms. This Agreement may only be modified by a writing signed by an authorized officer of Nokia, although Nokia may vary the terms of this Agreement.

Without restricting any rights of Nokia, clauses 4, 7, 9 and 10 confer a benefit on Symbian and are intended to be enforceable by Symbian. Subject only to the exception in this Clause, nothing in this Agreement confers or purports to confer on any third party any benefit or right to enforce any of the terms of this Agreement.

This is the entire agreement between Nokia and You relating to the Software, and it supersedes any prior representations, discussions, undertakings, end-user agreements, communications, or advertising relating to the Software.

PLEASE SUBMIT ANY ACCOMPANYING REGISTRATION FORMS TO RECEIVE REGISTRATION BENEFITS WHERE APPLICABLE

6.2 Acknowledgements

6.2.1 Copyright notices

This material, including documentation and any related computer programs, is protected by copyright controlled by Nokia. All rights are reserved. Copying, including reproducing, storing, adapting or translating, any or all of this material requires the prior written consent of Nokia. This material also contains confidential information, which may not be disclosed to others without the prior written consent of Nokia.

Nokia operates a policy of continuous development. Nokia reserves the right to make changes and improvements to any of the products described in this document without prior notice.

6.2.2 Open source software

This product includes certain software originating from third parties that is subject to 1. the GNU Library/Lesser General Public License (LGPL) and/or 2. other different or additional copyright license, permission, disclaimers or notices containing obligation or

permission to provide the source code of such software with the binary / executable form delivery of the said software. The exact license terms of LGPL and said certain other licenses, as well as the required copyright and other notices, permissions and acknowledgements are reproduced in and available for you at http://www.forum.nokia.com/series60browser at http://www.forum.nokia.com/series60browser. The complete corresponding machine-readable copy of the source code of such software as well as the binary included in this product under the LGPL and said other licenses and permission are available at http://www.forum.nokia.com/series60browser at http://www.forum.nokia.com/series60browser. Please refer to the exact terms of the LGPL and the other licenses regarding your rights under said licenses

This product includes API header files originally developed by Netscape Communications Corporation, modified by Nokia by changing certain variables for the purpose of porting the file to Symbian platform on May 1, 2004, and made available in source code form under the Netscape Public License v. 1.1 http://www.forum.nokia.com/browserapi/ at http://www.forum.nokia.com/browserapi.

6.2.3 Trademarks

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation.

S60 and logo is a trademark of Nokia Corporation.

Symbian, the Symbian logo, the Symbian 'i' and all Symbian-based marks are trademarks of Symbian Ltd.

These trademarks, and others, are acknowledged. Conventional abbreviations of these trademarks are used throughout the product documentation.

ARM, Thumb, StrongARM and ARM Powered are registered trademarks of ARM Limited.

ActiveState, ActivePerl, and PerlScript are trademarks of ActiveState Tool Corp.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc.

Microsoft®, Microsoft® Windows NT®, Microsoft® Windows® 95, Microsoft® Windows® 2000, Microsoft® Windows® XP, Microsoft® Visual C++®, Microsoft® Developer Studio™, Microsoft® Visual Studio™ are trademarks or registered trademarks of Microsoft Corporation.

Other product and company names mentioned herein may be trademarks or tradenames of their respective owners.

6.2.4 Third party copyright notices

Incorporates MPPC® compression from Hi/fn™.Stac ®, LZS ®, ©1996, Stac, Inc., ©1994-1996 Microsoft Corporation. Includes one or more U.S. Patents: No. 4701745, 5016009, 5126739, 5146221, and 5414425. Other patents pending.

Incorporates LZS® compression from Hi/fn™. Hi/fn ®, LZS ®,©1988-98, Hi/fn. Includes one or more U.S. Patents: No. 4701745, 5016009, 5126739, 5146221, and 5414425. Other patents pending.

Incorporates BSAFE cryptographic or security protocol software from RSA Security. Copyright © 1992-1999 RSA Security, Inc. All Rights Reserved.

Incorporates parts of the C Standard library code, Copyright © 1980, 1982, 1983, 1986, 1988, 1990, 1993 The Regents of the University of California. All rights reserved. Licence text.

US Patent No 5818437 and other pending patents. T9 text input software copyright (c) 1997-2002. Tegic Communications, Inc. All rights reserved.

Copyright 1996-2000, ActiveState Tool Corp. All rights reserved. Further text.

The following software is distributed with the SDK under the terms of the GNU General Public License, Copyright © 1989, 1991 Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA. Licence text.

6.2.5 FreeBSD Copyright Message

Copyright (c) 1980, 1982, 1983, 1986, 1988, 1993, 1998, 1990.

The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3 All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed by the University of California, Berkeley and its contributors.

4 Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.2.6 ActiveState Licence

ActiveState Perl

ActiveState, ActivePerl, and PerlScript are trademarks of ActiveState Tool Corp.

Commercial support for ActivePerl is available through the PerlClinic at http://www.PerlClinic.com. Peer support resources for ActivePerl issues can be found at the ActiveState Web site under support at http://www.activestate.com/support/.

The ActiveState Repository has a large collection of modules and extensions in binary packages that are easy to install and use. To view and install these packages, use the Perl Package Manager (PPM) which is included with ActivePerl.

ActivePerl is the latest Perl binary distribution from ActiveState and replaces what was previously distributed as Perl for Win32. The latest release of ActivePerl as well as other professional tools for Perl developers are available from the ActiveState website at http://www.ActiveState.com .

ActiveState Community Licence

Preamble:

The intent of this document is to state the conditions under which Packages may be copied and distributed, such that ActiveState maintains control over the development and distribution of the Package, while allowing the users of the Package to use the Package in a variety of ways. The Package may contain software covered by the Artistic License. The ActiveState Community License complies with Clause 5 of the Artistic License and does not limit your rights to software covered by the Artistic License.

Authorized electronic distributors of the Package are:

- Any ActiveState.com site
- Any complete, publicly available CPAN mirror listed at Perl.com at http://www.perl.com.

For more information on CPAN please see Perl.com at http://www.perl.com

Definitions:

"ActiveState" refers to ActiveState Tool Corp., the Copyright Holder of the Package.

"Package" refers to those files, including, but not limited to, source code, binary executables, images, and scripts, which are distributed by the Copyright Holder.

"You" is you, if you're thinking about copying or distributing this Package.

1 You may use this Package for commercial or non-commercial purposes without charge.

2 You may make and give away verbatim copies of this Package for personal use, or for use within your organization, provided that you duplicate all of the original copyright notices and associated disclaimers. You may not distribute copies of this Package, or copies of packages derived from this Package, to others outside your organization without specific prior written permission from ActiveState (although you are encouraged to direct them to sources from which they may obtain it for themselves).

3 You may apply bug fixes, portability fixes, and other modifications derived from ActiveState. A Package modified in such a way shall still be covered by the terms of this license.

4 ActiveState's name and trademarks may not be used to endorse or promote packages derived from this package without specific prior written permission from ActiveState.

5 THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

6.2.7 GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software — to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s

software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1 This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2 You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3 You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4 You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form)

with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5 You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6 You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7 Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8 If, as a consequence of a court judgement or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9 If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution

limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10 The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11 If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

12 **NO WARRANTY**

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING.

13 **REPAIR OR CORRECTION**

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

-- START COPY HERE --

<one line to give the program’s name and a brief idea of what it does.> Copyright © 19yy <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

-- END OF COPY--

Also add information on how to contact you by electronic and paper mail. If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

-- START COPY HERE --

Gnomovision version 69, Copyright © 19yy <name of author>

Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

-- END OF COPY--

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

-- START COPY HERE --

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker. <signatur of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

-- END OF COPY--

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.