



**Series 40 5th Edition SDK**

**User's Guide**

June 2007

---

Copyright © Nokia 2007. All rights reserved.

This document is for use with the Series 40 5th Edition SDK. Reproduction, transfer, distribution or storage of part or all of the contents in this document in any form without the prior written permission of Nokia is prohibited.

Nokia, Series 40 5th Edition SDK, and the Nokia Connecting People logo are trademarks or registered trademarks of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.OpenSSL.org/>).

Nokia operates a policy of on-going development. Nokia reserves the right to make changes and improvements to any of the products described in this document without prior notice.

UNDER NO CIRCUMSTANCES SHALL NOKIA BE RESPONSIBLE FOR ANY LOSS OF DATA OR INCOME OR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES HOWSOEVER CAUSED.

THE CONTENTS OF THIS DOCUMENT ARE PROVIDED "AS IS". EXCEPT AS REQUIRED BY APPLICABLE LAW, NO WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE MADE IN RELATION TO THE ACCURACY, RELIABILITY OR CONTENTS OF THIS DOCUMENT. NOKIA RESERVES THE RIGHT TO REVISE THIS DOCUMENT OR WITHDRAW IT AT ANY TIME WITHOUT PRIOR NOTICE.

The availability of particular phone products may vary by region. Please check with the Nokia dealer nearest to you.

Visit Forum Nokia (<http://www.forum.nokia.com>), the site designed for developers using technologies supported by Nokia.

---

# Contents

<b>About this User's Guide .....</b>	<b>7</b>
Who should read this guide?.....	7
Typographical conventions used in this guide .....	7
Other product documentation .....	8
<b>1 What is the Series 40 5th Edition SDK? .....</b>	<b>9</b>
Quick tour of the Series 40 5th Edition SDK .....	11
Getting information about the SDK content .....	13
About the Diagnostics window .....	13
About the tracing options for MIDlets .....	14
Setting up the Series 40 5th Edition SDK .....	15
Starting the Series 40 5th Edition SDK within an IDE .....	16
Getting information about the Series 40 5th Edition SDK .....	16
Sending comments to Nokia Corporation .....	17
Exiting from the Series 40 5th Edition SDK .....	17
What's next? .....	17
<b>2 Setting Up the Series 40 5th Edition SDK .....</b>	<b>19</b>
Displaying the SDK on top of other windows .....	19
Setting the general behaviors of the SDK .....	20
Setting the startup behaviors of the SDK .....	20
Changing the instance number prefix (telephone number) of the SDK .....	20
Changing the date and time of the SDK .....	20
Analyzing content on the SDK .....	21
Selecting the language of the handset menus .....	21
Setting the display behaviors of the SDK .....	21
Emulating a memory card .....	21
Removing the memory card .....	22
Adding the memory card .....	22
Playing phone sounds .....	22
Using the numeric keypad .....	22
Connecting a smart card reader .....	22
Setting up network access .....	23
Setting up a proxy server .....	23
Disabling the HTTP cache .....	24
Changing the RMI registry port .....	24
Setting MIDP provisioning and execution options .....	25
Setting the provisioning mode .....	25
Provisioning MIDlets (validating MIDlets through JAM) .....	25
Running MIDlets without JAM validation (direct loading) .....	26
Setting MIDlet execution options .....	27
Enabling or disabling user security prompts .....	27

---

Setting the KVM heapsize .....	27
Configuring the Persistent Record Management Store emulation .....	27
Setting up to reuse an SDK instance to run MIDlets .....	28
Invoking a MIDlet on the same SDK instance .....	28
Invoking a MIDlet on a new SDK instance .....	28
Setting MIDP tracing and speed options .....	28
Setting up tracing to file .....	28
Selecting tracing options .....	29
About MIDlet speed controls .....	30
Setting MIDlet canvas speed .....	30
Setting MIDlet KVM speed .....	30
What's next? .....	31
<b>3 Using the Series 40 5th Edition SDK .....</b>	<b>33</b>
About loading content on the SDK standalone application .....	34
Loading local content in the SDK menu .....	34
Loading URL content in the SDK menu .....	34
Loading URL content with the handset menus .....	35
Previewing and browsing with NMIT .....	35
Previewing WML files within Adobe GoLive CS2 .....	35
Showing a magnified view of the main window .....	36
Inputting text and commands from a keyboard .....	37
Emulating mouse clicks with keyboard shortcuts .....	37
Entering text from the keyboard .....	37
Button chording .....	38
Working with content types .....	38
What content types are mapped to what file types in the SDK? .....	38
What content types can you load directly using the browser? .....	40
What audio content types does the SDK support? .....	41
What video content types does the SDK support? .....	42
Generating events .....	42
What's next? .....	43
<b>4 Getting Information About the SDK's Content .....</b>	<b>45</b>
Working with the Diagnostics window .....	46
About the Traffic view .....	48
About the MIDP view .....	51
About KVM heapsize status .....	52
About the Browser Source view .....	52
About the Browser History view .....	56
About the Log view .....	57
Working with the PC file system .....	58
What's next? .....	60
<b>5 Simulating Wireless Connectivity between SDKs .....</b>	<b>61</b>
Working with the wireless simulation .....	62
Starting NCF .....	62
Shutting down NCF Lite 1.2 .....	62
What's next? .....	63

---

<b>6 Using the SDK for Message Content Development .....</b>	<b>65</b>
Working with MMS messages.....	66
Creating and sending an MMS message directly on the SDK.....	67
Creating and previewing an MMS message using NMIT .....	69
Processing an MMS message on the SDK .....	69
Working with DRM messages.....	70
Creating a DRM message in NMIT .....	71
Working with digital rights.....	73
Loading DRM rights on the SDK .....	74
Working with an SMS message .....	74
Creating an SMS message on the SDK.....	74
Receiving an SMS message on the SDK .....	74
Working with Bluetooth .....	75
Sending a file between SDK instances using Bluetooth .....	75
Receiving a file from an SDK instance through Bluetooth .....	75
Importing a UI theme.....	76
What's next?.....	76
<b>7 Using the SDK for MIDlet Development .....</b>	<b>77</b>
Setting up the SDK to run a MIDlet.....	78
About OTA provisioning and direct loading of a MIDlet.....	78
Direct loading of a MIDlet.....	79
OTA Provisioning a MIDlet .....	80
MIDP differences between the handset and the SDK .....	80
MIDlet connectivity differences .....	80
MIDP security differences.....	81
JSR 135 audio/video differences .....	81
Using the Series 40 5th Edition SDK with Eclipse 3.2.2 .....	81
Creating a new project for the MIDlet .....	81
Creating or importing classes for the MIDlet.....	82
Creating the MIDlet application package .....	82
Testing the MIDlet .....	82
Debugging the MIDlet .....	83
Using the Series 40 5th Edition SDK with Sun NetBeans.....	83
Additional information about MIDlets .....	84
What's next?.....	84
<b>8 Running the SDK from a Command Line.....</b>	<b>85</b>
Accessing the command line prompt.....	86
About the command line syntax .....	86
About the executables .....	87
About emulator.exe .....	87
About sdk.exe .....	87
About options.....	87
Getting Information about the CLI options .....	88
Options for managing SDK instances .....	88
Options for general use.....	89
Options for tracing Java runtime .....	90

---

Options for general management of Java output .....	91
About commands.....	93
About the Load command.....	93
About the Close command.....	94
-exit and -exitall options.....	94
-exit option .....	94
-exitall option.....	94
Working with a CLI shell to repeatedly send commands .....	95
<b>Index .....</b>	<b>97</b>

---

# About this User's Guide

This guide describes how to use the Series 40 5th Edition SDK.

## Who should read this guide?

You should read this guide, if you are a developer and plan to create content for mobile phone handsets that conform to the Series 40 Platform 5th Edition.

## Typographical conventions used in this guide

This guide uses the following typographical conventions:

Convention	Explanation
Courier	Text that you enter (as opposed to system prompts and responses); for example, file paths, commands, and program code
<i>Italic</i>	<ul style="list-style-type: none"><li>Names of books and documents</li><li>New terminology</li></ul>
<b>Bold</b>	Names of Windows menus, commands, buttons, and icons
<a href="#">URL link</a>	Active link

---

## Other product documentation

The following documents contain additional information about the SDK:

- *Series 40 5th Edition SDK Installation and Configuration Guide*
- *Series 40 5th Edition SDK Release Notes*

These documents are available in the zip package when the user's downloads it from Forum Nokia. Additional documents about this product and related technologies may also be available at [www.forum.nokia.com](http://www.forum.nokia.com).



---

# 1

## What is the Series 40 5th Edition SDK?

The Series 40 5th Edition SDK is a software tool that lets you test WML, XHTML, and MMS content and develop and debug Mobile Information Device Profile (MIDP) applications for use on Nokia mobile handsets. The SDK provides a development environment for media content and Java™ Platform, Micro Edition (Java™ ME) applications.

The Series 40 5th Edition SDK supports:

Area	Specification
Java APIs	JSR 75 File connection v1.0 and PIM v1.0 APIs
	JSR 82 APIs for Bluetooth v1.1
	JSR 118 MIDP v2.1
	JSR 120 Wireless Messaging API 1.1
	JSR 135 Mobile Media API 1.1 (audio and video play and audio capture only; does not support Real Time Streaming Protocol (RTSP), video capture, camera control, or FM radio)
	JSR 139 Connected Limited Device Configuration (CLDC) 1.1
	JSR 172 JAX-XML Web Services API (XML parsing) and JAX-RPC API
	JSR 177 Security and Trust Services APIs SATSA-APDU and SATSA-CRYPTO
	JSR 184 Mobile Graphics API 1.1
	JSR 185 Java Technology for Wireless Industry 1.0
	JSR 205 Wireless Messaging API 2.0
	JSR 226 Scalable 2D Vector Graphics API 1.0
	JSR 234 Advanced Multimedia Supplements (3D Audio & Music)
	JSR 248 Mobile Service Architecture Subset API 1.0
Browsing	XHTML Mobile Phone (MP) Profile
	User Agent Profile (UAProf) static
	Macromedia Flash Lite 2.1
Messaging	Multimedia Messaging Service (MMS) with SMIL support
Digital Rights Management	Open Mobile Alliance (OMA) Digital Rights Management (DRM) 1.0 (not for MIDlets)
Device Management	OMA Client Provisioning 1.1
	WAP/OMA Bootstrapping

If Nokia Connectivity Framework (NCF) is running, multiple instances of the SDK running on the same PC:

- Automatically recognize one another.
- Support the exchange of information through the Wireless Messaging API (GSM Short Message Service [SMS] format).
- Support standard SMS text messages. See [Working with an SMS message](#) on page 74.
- Support Bluetooth connectivity.

Without NCF, multiple instances of the SDK running on the same PC:

- Support the exchange of MMS messages. See [Creating and sending an MMS message directly on the SDK](#) on page 67.

Diagnostics functions include on/off tracing, Traffic and MIDP views, and an Event Generator that simulates unsolicited occurrences, such as an incoming phone call. The SDK features a color XHTML browser, MMS, and a TCP/IP protocol stack. The following three types of OMA-defined DRM 1.0 are supported:

- Forward Lock
- Combined Delivery
- Separate Delivery

The SDK is shipped with a set of standard languages. To add other languages, you can install World Languages packs.

## Quick tour of the Series 40 5th Edition SDK

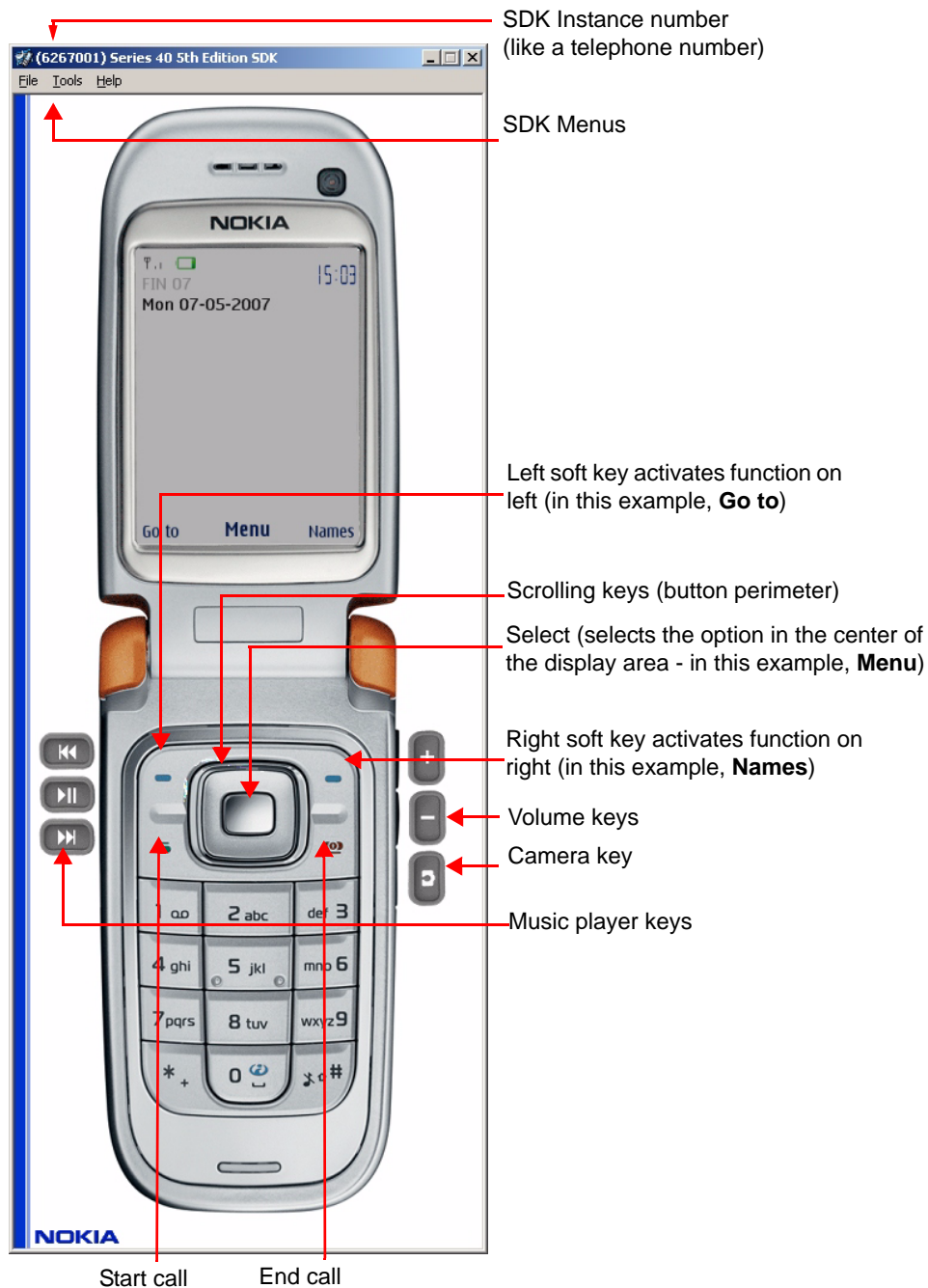
You can run the SDK:

- As a standalone application. Select **Start menu>Programs>Nokia Developer Tools>SDKs>Series 40>Series 40 5th Edition SDK**.
- In an integrated development environment (IDE) that lets you create content with supporting applications and immediately test the content with the SDK. See [Using the SDK for Message Content Development](#) on page 65 or [Using the SDK for MIDlet Development](#) on page 77.

For this quick tour, start the SDK as a standalone.

You can perform most functions on the SDK that are available on a handset and conform to the Series 40 Platform 5th Edition. Because the SDK is a developer's tool, you cannot make a phone call on the SDK. When you attempt to place a call on the SDK, the SDK simulates the call. Select the **End Call** key to cancel the call function, or the call will cancel automatically and the message, *Number not in use*, displays.

After starting the SDK, the main window looks like this:



The contrast and color rendering capabilities of the SDK and a handset may differ slightly because of the differences in monitor calibration, imaging technologies, and ambient lighting.

To simulate pressing the buttons on a mobile handset, click the corresponding key on the SDK.

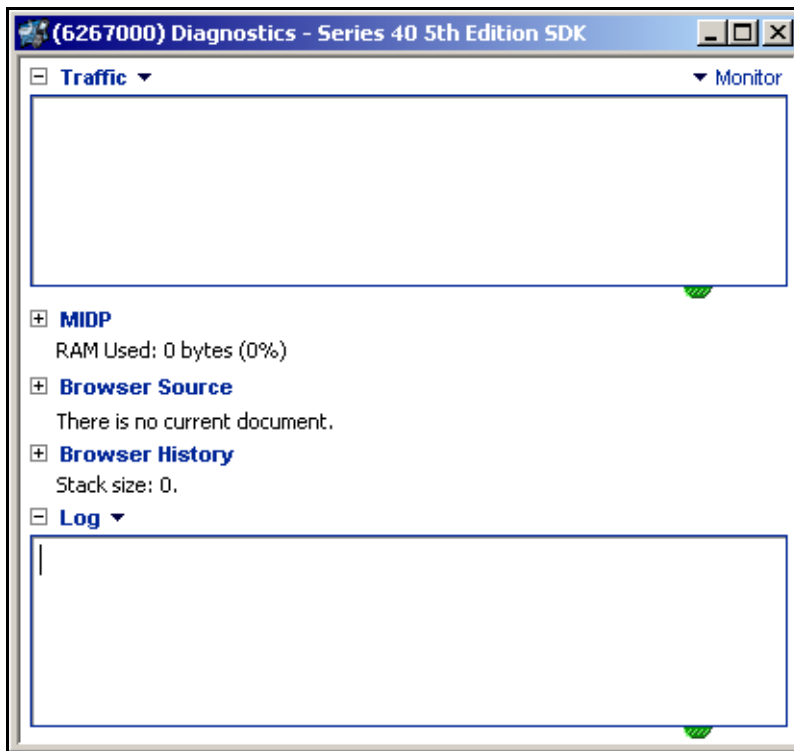
## Getting information about the SDK content

You can get information about the content on the SDK using:

- Diagnostics window - Provides information about how the content behaves. See [Getting Information About the SDK's Content](#) on page 45.
- Command line interface (CLI) tracing options - Helps you debug a MIDlet. See [Running the SDK from a Command Line](#) on page 85 and [About the tracing options for MIDlets](#) on page 14.

## About the Diagnostics window

When you select Tools>Diagnostics, the Diagnostics window appears:



The Diagnostics window allows you to view the following information:

<b>To see</b>	<b>Look at this view</b>
Information about requests to and responses from the Internet	Traffic
Information about KVM status	MIDP
Source code of the content that is currently loaded on the SDK browser	Browser Source
A history list of the SDK browser	Browser History
Messages generated by the SDK	Log

For more information about the Diagnostics window, see [Getting Information About the SDK's Content](#) on page 45.

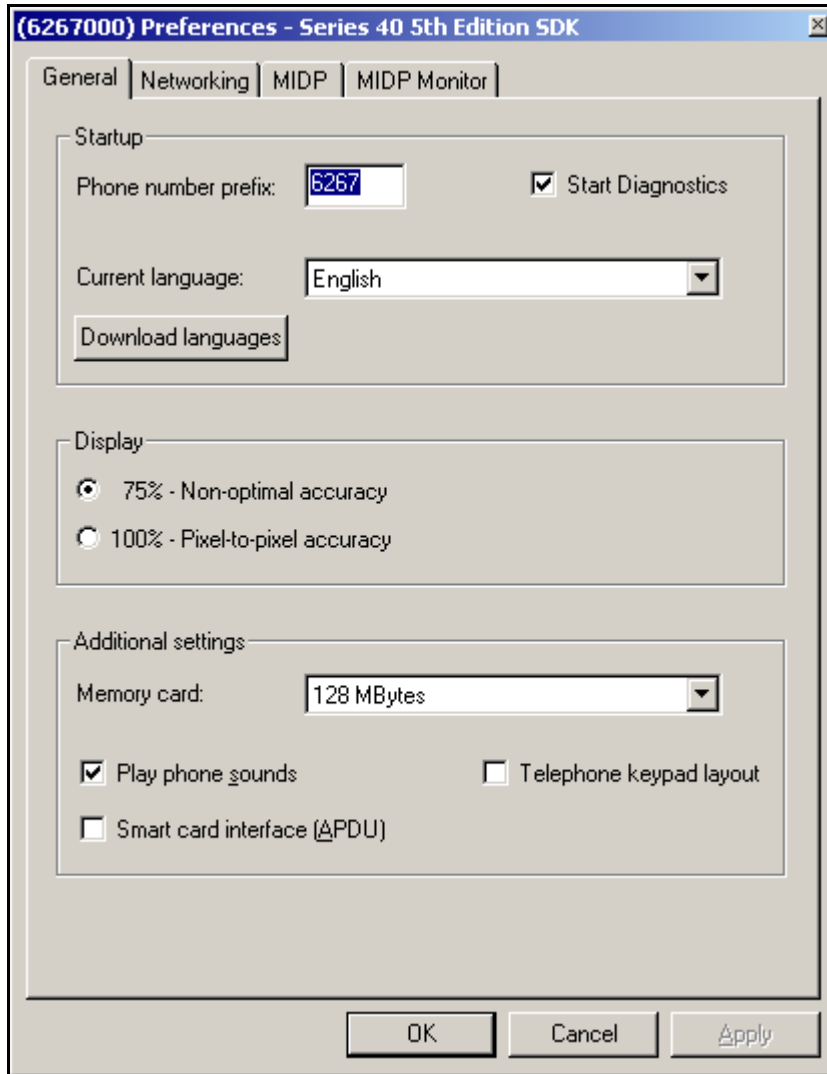
## About the tracing options for MIDlets

You can activate tracing options that help you debug the content on the SDK. The reports from the tracing options display in the DOS Console.

For more information about tracing options and running the SDK from the command line, see [Setting Up the Series 40 5th Edition SDK](#) on page 19 and [Running the SDK from a Command Line](#) on page 85.

## Setting up the Series 40 5th Edition SDK

To control aspects of how the SDK looks and operates, select **Tools>Preferences**. The Preferences window appears:



In Preferences, you can set values for attributes, such as the language in which the handset menus display and settings for running MIDlets. See [Setting Up the Series 40 5th Edition SDK](#) on page 19.

## Starting the Series 40 5th Edition SDK within an IDE

While you can use the SDK as a standalone application, you can also use it within an IDE to create a powerful development and testing environment. The following IDEs support the SDK:

Browser IDEs	Messaging IDEs	MIDP Development IDEs
Nokia Mobile Internet Toolkit 4.1 with a patch for JRE 5 (NMIT)	NMIT 4.1 with a patch for JRE 5	Eclipse 3.2.2 with EclipseME 1.6.6
Adobe GoLive CS2	Adobe GoLive CS2	Sun NetBeans 5.5 with NetBeans Mobility Pack 5.5

Adobe Dreamweaver 8

You might need to configure your IDE to recognize the SDK as a supported device before you can launch the SDK from within the IDE interface. For information about configuring IDEs to work with the SDK, select **Help>Installation and Configuration Guide**.

## Getting information about the Series 40 5th Edition SDK

The Help menu provides useful information along with a search option. You can get information about the SDK from the **Help** menu:

For information on	Select
Using the SDK	Help>SDK Help
Late-breaking news	Help>Release Notes
Java API	Help>SDK Help>Java API Documentation
Sample Java MIDlets	Help>Midlet Samples
Developer resources	Help>SDK Help>Developer resources
Registering this product	Help>Register Now...
Version of this SDK	Help>About



## Sending comments to Nokia Corporation

You can send any comments you have about the SDK to Nokia Corporation, if you have an active Internet connection. Select **Help>Provide feedback to Nokia**. Selecting this option automatically opens <http://www.forum.nokia.com/feedback>, where you can enter your comments.

## Exiting from the Series 40 5th Edition SDK

Select **File>Exit**. The SDK terminates.

## What's next?

If you want to begin using the SDK as a standalone application right now, you can:

- Set up how the SDK operates and looks. See [Setting Up the Series 40 5th Edition SDK](#) on page 19.
- Begin to load content on the SDK to see how the content looks and works. See [Using the Series 40 5th Edition SDK](#) on page 33.

Or, you might want to explore running the SDK as a standalone application from a command line. See [Running the SDK from a Command Line](#) on page 85.



---

# 2

## Setting Up the Series 40 5th Edition SDK

You can control how the SDK operates and looks by modifying:

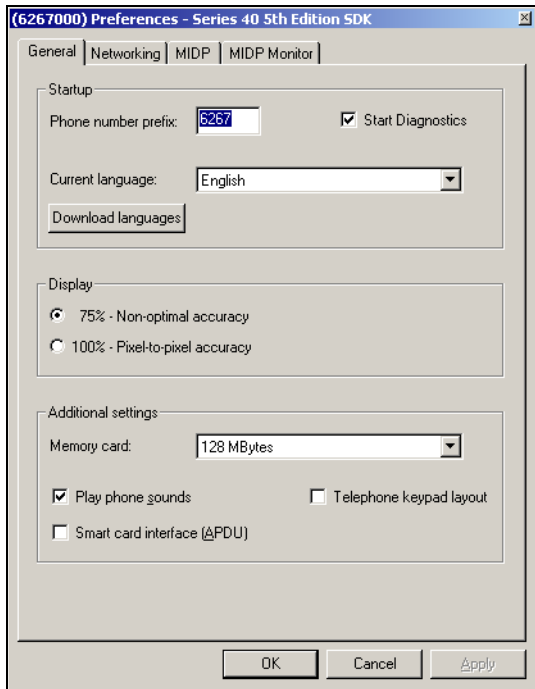
- Appearance of the SDK main window, the language of the menus in the main window, and the audio behavior of the SDK. You can also select to connect a smart card reader to the SDK.
- Networking settings, such as whether to use a network proxy.
- MIDlet setup, such as provisioning and execution options, tracing options, KVM and canvas speed controls, and whether to start a new SDK whenever you launch a MIDlet.

### Displaying the SDK on top of other windows

To display the SDK on top of all other windows, select **Tools>SDK on Top**. When the setting is activated, a check mark appears next to the menu selection.

## Setting the general behaviors of the SDK

Select **Tools>Preferences** and click the **General** tab to set up how the SDK appears and operates. View the **General** tab:



## Setting the startup behaviors of the SDK

You can change the startup behavior of the SDK. These settings (except for the memory card setting) take effect the next time you start a new instance of the SDK.

### Changing the instance number prefix (telephone number) of the SDK

You can change the first four numbers of the SDK's instance number (default is 6267) by entering the number in **Phone number prefix**. The last three numbers are automatically generated by the SDK. For example, if you enter the numbers 2345 as the phone number prefix, the next SDK instance you start will have a telephone number of 2345000.

### Changing the date and time of the SDK

Every time you start the SDK, the emulator emulates the behavior of the handset when first switched on by briefly displaying the pop-up window for entering the time. In the handset, this dialog is displayed when the device is turned on for the first time or when the date and time information have become unavailable because of, for example, the battery running out. However, the SDK automatically retrieves the time and date information from the computer's operating system while starting.

If you want to change the time and date settings, from the SDK handset, select **Menu>Settings>Time and date**.

## Analyzing content on the SDK

You can check **Start Diagnostics** to activate the Diagnostics window to view information that can help you analyze and troubleshoot the content on the SDK. If you do not check this option, the Diagnostics window does not appear when you select **Tools>Diagnostics**. Disabling the Diagnostics window makes the SDK start faster. For more information about the Diagnostics window, see [Getting Information About the SDK's Content](#) on page 45.

## Selecting the language of the handset menus

For localization, you can select the language for the menus within the SDK's LCD simulation. This setting does not affect the menus in the SDK menu bar, which remain in English. In addition, it does not affect downloaded content, which remains in the language in which the documents were created.

You can click **Download Languages** to download and install additional languages. This selection takes you to a page on the Nokia website where you download the language pack you want. For more information about languages, see *Series 40 5th Edition SDK Installation and Configuration Guide*.

## Setting the display behaviors of the SDK

Select one of the following image sizes for the main window of the SDK:

- **75% - Non-optimal accuracy**  
This option provides a complete view of the handset with the fold open. The numeric keypad of the handset is visible.
- **100% - Pixel-to-pixel accuracy**  
This option displays the front cover of the handset (the screen, call keys, navigation keys, soft keys and the music player and camera keys). You can access the numeric keypad by scrolling down in the SDK window.

**Note:** By default, the SDK uses the 75% option. If you want to achieve pixel-to-pixel accuracy, for example to test the placement of pixels on the handset screen, select the 100% - **Pixel-to-pixel accuracy** option.

## Emulating a memory card

A Nokia 6267 handset can use a memory card to augment its memory. The SDK emulates this memory card in its file system emulation. See [Working with the PC file system](#) on page 58.

## Removing the memory card

To emulate removing a memory card, use the drop-down menu beside **Memory card** to select **Card removed**. You can emulate removing the memory card, while the SDK is running, just as you can remove the memory card from the phone, while it is running.

## Adding the memory card

A Nokia 6267 handset supports different memory card sizes, while the size of the emulated memory card in the SDK is 128 Mbytes. To emulate a memory card, use the drop-down menu beside **Memory card** to select **128 MBytes**. By default, the memory card is selected for each new SDK instance.

## Playing phone sounds

Check **Play phone sounds** to play supported sound content. If you leave this box unchecked, you won't hear any sound from the SDK.

## Using the numeric keypad

Computer and phone numeric keypads are reversed. Computers have numbers seven, eight, and nine on the top row and phones have seven, eight, and nine on the third row. By default, the SDK uses the computer keypad layout, which means typing a value enters the same value into the SDK but results in the wrong cursor movement for applications that use the keypad for movement.

Select **Telephone keypad layout** to use the correct cursor movement. However, typing the values 123456789 will result in the wrong values on the SDK, for example, 789456123.

The behavior of the numeric keys on the main part of the keyboard is not affected by the telephone keypad layout setting.

## Connecting a smart card reader

The SDK supports a smart card reader connection (JSR 177).

If you want to connect a smart card reader to the SDK, check **Smart card interface (APDU)** and connect the smart card reader to your PC. Next, select **OK**. A confirmation window appears. Select **OK**. Restart the SDK. On start-up, a pop-up window appears with the following information:

- If there is only one smart card reader connected to your PC, the options are either **No reader attached** or the name of the reader you have attached.
- If there are more than one smart card readers connected to your PC, all the readers are listed. You can select only one reader at a time from the list. After selecting the reader, the options are either **No reader attached** or the name of the reader you chose from the list.

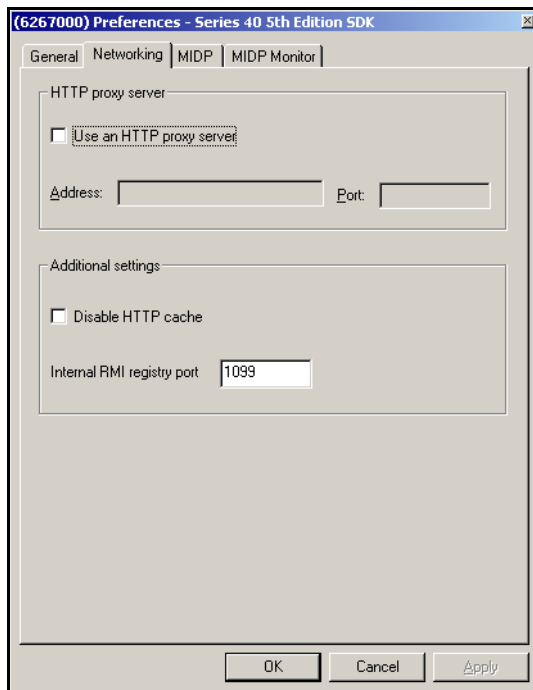
**Note:** If you have not checked the **Smart Card interface (APDU)** option, the SDK does not recognize any smart card readers that you have connected to your PC.

**Note:** Insert the smart card into the reader before SDK start-up. The smart card interface is cold swappable.

If you experience problems with the smart card reader connectivity, note that you can redirect the possible errors to a chosen log file of the SDK. This log file is not automatically generated. To capture the smart card reader information in a file, the SDK instance and the monitoring to file must be started in the CLI, using the output redirection functionality. To start the logging, use the following: `emulator > log.061208`. Note that you can use the same log file for MIDP tracing.

## Setting up network access

Select **Tools>Preferences** and click the **Networking** tab to set up a proxy server or to change the RMI Registry port. View the **Networking** tab:



### Setting up a proxy server

The SDK uses the HTTP protocol to browse content, load MIDlets, and allow MIDlets to communicate with one another. HTTP requests from your computer might pass through a proxy, if you are working within a Corporate Intranet. In this case, you'll have to specify an HTTP proxy for the SDK to use.

To specify the address of the proxy server (for example, 113.14.176.11 or proxy.domain) and a port number, check **Use an HTTP proxy server** and enter the information. This information is available from your network administrator.

## Disabling the HTTP cache

Nokia 6267 handsets have an HTTP cache that stores downloaded Internet content. This feature improves browser performance. To enable the cache in the SDK, uncheck **Disable HTTP cache**, which causes the SDK to behave like the actual handset.

Unlike the handset, the SDK lets you disable this cache. Check the box, if you prefer to retrieve content over the Internet each time you browse to a location. This option allows you to view any changes you've made to the content on an Internet Web server and ignores all cache directives from previous downloads.

## Changing the RMI registry port

The SDK requires an internal port for access to the RMI registry, which defaults to port 1099. If another application is already using that port, a conflict arises that might cause the SDK or the application using the port to freeze or might prevent the SDK from interoperating with other tools. This conflict rarely occurs and when it does, a message with instructions opens when the SDK starts.

If this conflict occurs, you may want to resolve it by rebooting the computer. If this conflict happens frequently, you can change the internal RMI registry port number.

To find out which port numbers are not in use, go to **Start menu>Run** and enter `CMD` to open a command line. At the command line, enter `netstat -a`. A list of running applications and the ports they use opens.

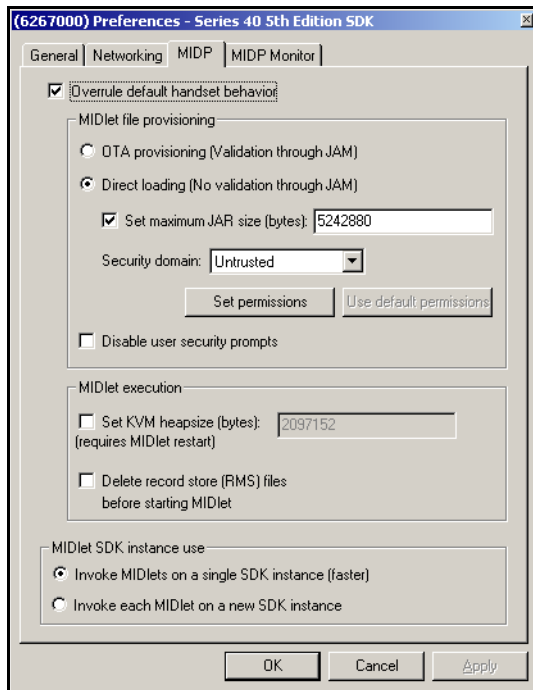
Select **Tools>Preferences** and click the **Networking** tab. Enter a port number that is not being used for the **Internal RMI registry port**.

After you enter a new port number, you'll have to reboot the computer for the change to take effect. If you do not reboot, currently running SDK instances will use the old port number and newly launched SDK instances will use the new port number. The currently running and the new SDK instances will not work together.



## Setting MIDP provisioning and execution options

Select **Tools>Preferences** and click the **MIDP** tab to set options that help you run MIDlets. View the **MIDP** tab:



To access the **MIDlet file provisioning** and the **MIDlet execution** options on the **MIDP** tab, check **Override default handset behavior**. The **MIDlet SDK instance use** options are always available.

## Setting the provisioning mode

The default behavior of the SDK loads MIDlets directly. For information about loading MIDlets with and without provisioning, see [About OTA provisioning and direct loading of a MIDlet](#) on page 78.

### Provisioning MIDlets (validating MIDlets through JAM)

To provision a MIDlet, check **OTA Provisioning** in the **MIDlet File Provisioning** section of this tab. The MIDlet is then loaded through the Java Application Manager (JAM), which validates that the Java Application Descriptor (JAD), Java ARchive (JAR), and JAR manifest in the JAR file that make up a MIDlet are correct. When you load a JAD file with provisioning enabled, the JAR and JAR manifest associated with it are validated. When you provision a JAR file, only the JAR manifest is validated.

## Running MIDlets without JAM validation (direct loading)

Select **Direct loading** to run a MIDlet without validating it through the JAM. This option is the default value.

When you select **Direct loading**, you can also:

- Set the maximum JAR file size of a MIDlet that the SDK can run. The maximum JAR file size you can enter is 5,242,880 bytes (5 megabytes). The maximum JAR file size that the Nokia 6267 handset can use is 1,048,576 bytes (1 megabyte).
- Select the type of security domain you want the SDK to apply to your MIDlet and modify the permissions associated with that domain.

To set the network security level:

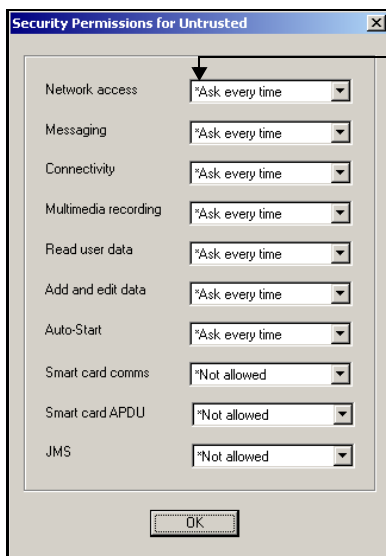
- 1 Select the security domain from the **Security Domain** menu from the following options:

- **Trusted**
- **Untrusted**
- **Maximum**
- **Minimum**

The **Trusted** and **Untrusted** domains are defined in the MIDP 2.0 standard and are available in the handset. The SDK provides the additional **Maximum** and **Minimum** domains to simplify MIDlet debugging.

**Note:** To run MIDlets without any security restrictions, select **Maximum**, which sets all permissions to **Always Allowed**. The **Minimum** domain sets all permissions to **Not Allowed**.

- 2 Click **Set Permissions** to view or modify the active settings of each permission group for the security domain you've selected. The following **Security Permissions for Untrusted** window opens:



Asterisk indicates the value is a default for the security domain.

- 3 In the **Security Permissions** window, modify the permissions for each function group by making selections from the menus.

The default permissions for each function group in the domain are marked with an asterisk. Permissions other than default that you select are not marked with an asterisk.

When you change a default permission, **Use default permissions** is highlighted in the **General** tab of the Preferences window. You can click **Use default permissions** to reset the permissions for all the function groups in the chosen domain to the default values.

## Setting MIDlet execution options

In the **MIDlet provisioning** section of this tab, you can configure how a MIDlet executes at runtime.

### *Enabling or disabling user security prompts*

When you check the **Disable User Security Prompts** option, the SDK stops displaying security prompts when you run MIDlets. Checking this box is equivalent to changing all **Ask first** and **Ask every time** permissions to **Allowed**.

### *Setting the KVM heapsize*

The size of the KVM memory heap (heapsize) controls how much memory the MIDlet can take to run. The maximum heapsize you can enter is 10,485,760 bytes (10 megabytes). The maximum heapsize for the Nokia 6267 handset is 2,097,152 bytes (2 megabytes).

You can simulate the memory constraints of different devices. Let's say you are working on a handset that has a maximum heapsize of 320K. You may want to adjust the heapsize on the SDK to 320K to verify that the MIDlet can run in this memory space. To set a new value for the KVM heapsize, check **Set KVM heapsize** and specify a new value. The new KVM heapsize takes effect the first time you run a MIDlet after you set the heapsize. See `-xheapsize` option in [Options for general use](#) on page 89.

### *Configuring the Persistent Record Management Store emulation*

A Record Management System (RMS) file is a file a MIDlet creates that remains after the MIDlet ends. For example, an RMS file from a MIDlet might contain scores from the last time you played a game. By default, the SDK saves RMS files in the PC file system, making them persistent. To simulate the behavior of a newly installed MIDlet, you can remove any old RMS files from the SDK by checking **Delete record store (RMS) files before starting MIDlet**.

## Setting up to reuse an SDK instance to run MIDlets

You can select the reusing of the same SDK instance to run all the MIDlets you launch, instead of launching a new SDK each time you invoke a MIDlet. Reusing an SDK instance is faster than starting a new instance every time you want to run a MIDlet. However, starting a new SDK each time you invoke a MIDlet allows you to compare and communicate between MIDlets.

### *Invoking a MIDlet on the same SDK instance*

To reuse the same SDK instance every time you invoke a MIDlet, select **Invoke MIDlets on a single SDK instance (faster)**. If more than one SDK is running, this option uses the latest SDK instance that was created (typically, the one with the highest number).

When you select this option, verify that the following message opens in the Windows console before you send a MIDlet to the SDK:

```
Nokia Series 40 5th Edition SDK Instance #6267000 Ready for Future
Connections
```

If you try to invoke a MIDlet too soon after making this selection, a second SDK instance is started.

### *Invoking a MIDlet on a new SDK instance*

To start a new SDK instance each time you invoke a MIDlet, select **Invoke each MIDlet on a new SDK instance**. This option comes in handy when you want to make a side-by-side comparison of how two MIDlets run.

Even if you use the **Invoke MIDlets on a single SDK instance (faster)** option, you can always start a new SDK instance manually without having to change that option. Perform *one* of the following actions:

- Select **Start menu>Programs>Nokia Developer Tools>SDKs>Series 40>Series 40 5th Edition SDK**.
- In the command line, enter `emulator -Xnew`.

## Setting MIDP tracing and speed options

To achieve the best performance in MIDlet development, the SDK allows you to set up MIDP tracing and MIDlet speed for monitoring your MIDlet applications.

### Setting up tracing to file

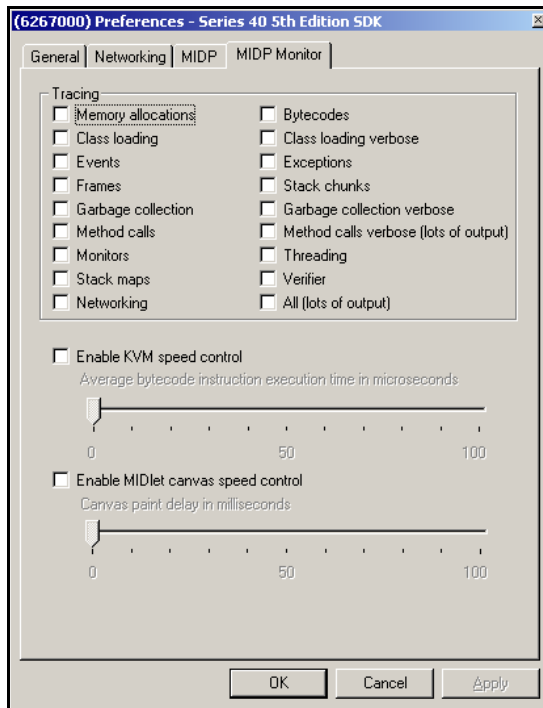
The MIDP tracing is handled by redirecting the output of the SDK to a log file. To capture the tracing information in a file, monitoring to file must be started in the CLI, using the output redirection functionality. Although you can select what to trace from the SDK Preferences, you cannot start the tracing to file from the SDK's Preferences dialog. The log file has to be created first.

To start the tracing, you can use, for example, the following: in the `bin` folder, enter

`emulator > o:\log.6267`. This starts an SDK instance, which prints the output to a file named `log.6267` in o-drive.

Next, in the SDK, select **Tools>Preferences** and click the **MIDP Monitor** tab to view options that let you trace the performance of a MIDlet and control the speed at which it runs. These tracing options can also be set in the CLI. See [Options for tracing Java runtime](#) on page 90.

View the MIDP Monitor tab:



## Selecting tracing options

Either from the Preferences or in the CLI, set the tracing options you want, to capture to the log file. These tracing options trace events that occur in the KVM. (To view events that occur between the SDK and an external entity, such as a gateway simulator or Internet, see [About the Traffic view](#) on page 48.)

To view the log file, you can use, for example, the standard text editor which is provided with your PC's operating system. Note that the log file exists for viewing purposes of the SDK's output stream only, and editing it does not affect the SDK behaviour or the MIDlets.

To use tracing options within an IDE, you must set the tracing options within the IDE. Using tracing options and breakpoints within an IDE lets you trace specific sections of code in a MIDlet. To see if the IDE you are using supports tracing, refer to the IDE documentation.

**Note:** Selecting too many tracing options can slow down the SDK.

## About MIDlet speed controls

When you run a MIDlet on a fast computer, you might not get an accurate demonstration of how the MIDlet runs on a handset, because the simulation usually runs faster on the computer than it does on the handset. MIDlet speed controls (KVM and Canvas speed) let you slow down the performance of a MIDlet on a computer to better reflect the MIDlet's performance on a handset. Developing a MIDlet in an environment with a slower KVM or canvas delay speed can help you monitor and optimize your code.

However, turning on the canvas speed and KVM speed simultaneously can adversely affect the data you collect, because canvas speed and KVM speed both control the speed of the SDK executable in ways that can conflict. For example, let's say you are testing a MIDlet game with an intricate graphical display that requires frequent updating, and you slow down the canvas speed to better simulate the graphical display on a handset. Simultaneously slowing down the KVM speed will additionally slow down the canvas speed, so the performance of the MIDlet will be distorted and the data you receive will be inaccurate.

Adjusting the KVM speed control affects the canvas speed, but adjusting the canvas speed control does not affect the KVM speed.

### *Setting MIDlet canvas speed*

Use the MIDlet canvas speed delay slider to slow down a MIDlet that displays too rapidly on a fast computer. This setting is recommended for a MIDlet with an active graphical display that requires frequent updating (for example, a game).

Move the slider to the right to increase the delay in milliseconds of the canvas paint function. Because the maximum delay you can set is 100 milliseconds, this setting has no visible effect, unless the canvas repaint rate of an object is significantly greater than 10 times per second.

### *Setting MIDlet KVM speed*

Use this slide setting to regulate the speed of the bytecode instructions in the KVM, which is the speed at which a MIDlet runs. This setting is recommended for a MIDlet whose graphical display does not need to be updated frequently (for example, an application that calculates a mathematical equation and displays the result).

While this setting can be helpful, the KVM speed control does not provide a reliable way to gauge the actual speed of a MIDlet on a handset. To be certain, test the MIDlet on a handset.

A value of 0 (zero) indicates this feature is turned off and that the KVM is running at maximum speed. The maximum delay is 100 microseconds.

## What's next?

Now that you've customized the SDK, you can go ahead and begin:

- Displaying wireless content, as described in [Using the Series 40 5th Edition SDK](#) on page 33.
- Developing messages and browser content, as described in [Using the SDK for Message Content Development](#) on page 65.
- Developing MIDlets, as described in [Using the SDK for MIDlet Development](#) on page 77.

Or, you might want to explore running the SDK as a standalone application from a command line. See [Running the SDK from a Command Line](#) on page 85.





---

# 3

## Using the Series 40 5th Edition SDK

This chapter describes how you can perform the following tasks using the SDK to develop wireless content:

- Displaying content to check the appearance and test its function.
- Entering commands and text in response to the content.
- Testing the content under the influence of unpredictable events.

## About loading content on the SDK standalone application

To check the appearance and function of your wireless content on the SDK, perform *one* of the following actions:

- Open a local file using **File>Open**.
- Open a URL using **File>Open URL** or by clicking the SDK keys to navigate to a website.

**Note:** You may have to set up a proxy server. See [Setting up network access](#) on page 23.

**Note:** You can also use the **Open URL** command to open a file in your PC.

- Open a recently used file or URL by selecting its name in the **File** menu.

**Note:** If you open a URL, you may have to set up a proxy server. See [Setting up network access](#) on page 23.

- Load content from a supporting application within an IDE. NMIT, Adobe GoLive CS2 and Adobe Dreamweaver let you display content on the SDK that you've developed with their editors. You can view the content by either clicking a button or making a menu selection in the application.
- Use the handset's web browser to access wireless content: select **Menu>Web>Go to address** and enter the web address.

**Note:** Series 40 5th Edition SDK supports Internet Protocol version 4 (IPv4), while the actual Series 40 5th Edition handsets support also Internet Protocol version 6 (IPv6).

### Loading local content in the SDK menu

To load local content into the SDK when you are using the SDK as a standalone application:

- 1 Select **File>Open**. The **Open** dialog box opens.
- 2 Navigate to the local file or enter its full path name and click **OK**. The SDK loads the content.

### Loading URL content in the SDK menu

To load URL content into the SDK when you are using the SDK as a standalone application:

- 1 Select **File>Open URL**. The **Open** dialog box opens.
- 2 Enter a URL and click **OK**. The SDK loads the content.

**Note:** You can also open a local file by entering the full path name of the file or click **Browse** to navigate to it. If you start the file name with `sdk : / /`, you can use the relative path of the file (this way the system will use the installation directory of the SDK as the root directory), or the absolute path. You can use both `/` and `\` in the file paths.

## Loading URL content with the handset menus

You can also load a website using the menus on the SDK's handset. For example:

- 1 Use the soft keys to select **Menu>Web>Go to address**.
- 2 Enter a URL, such as: `http://www.vertu.com`.
- 3 Use the soft keys to select **OK**.

## Previewing and browsing with NMIT

You can preview local files and browse Internet content on the SDK when you use the NMIT.

- Select **File>Open** to open a local file in NMIT's Editor window. Click **Show** to preview the file on the SDK.
- Select **File>New** to create a new file in NMIT's Editor window. Click **Show** to preview the file on the SDK.

Mobile internet content

- Select **Tools>SDK Control Panel**, if the panel is not visible. Enter the URL you want in the address bar and press the **Enter** key.

## Previewing WML files within Adobe GoLive CS2

When you use Adobe GoLive, you can preview only local content on the SDK. You cannot browse mobile Internet content from within GoLive.

To preview local content:

- 1 Perform *one* of the following actions in Adobe GoLive CS2:
  - Select **File>New Special>WML Deck** to create a new WML file in the GoLive's Editor window.
  - Select **File>Open** to open an existing WML file in the GoLive's Editor window.
- 2 Select **File>Preview In>emulator.exe** to display it on the SDK.

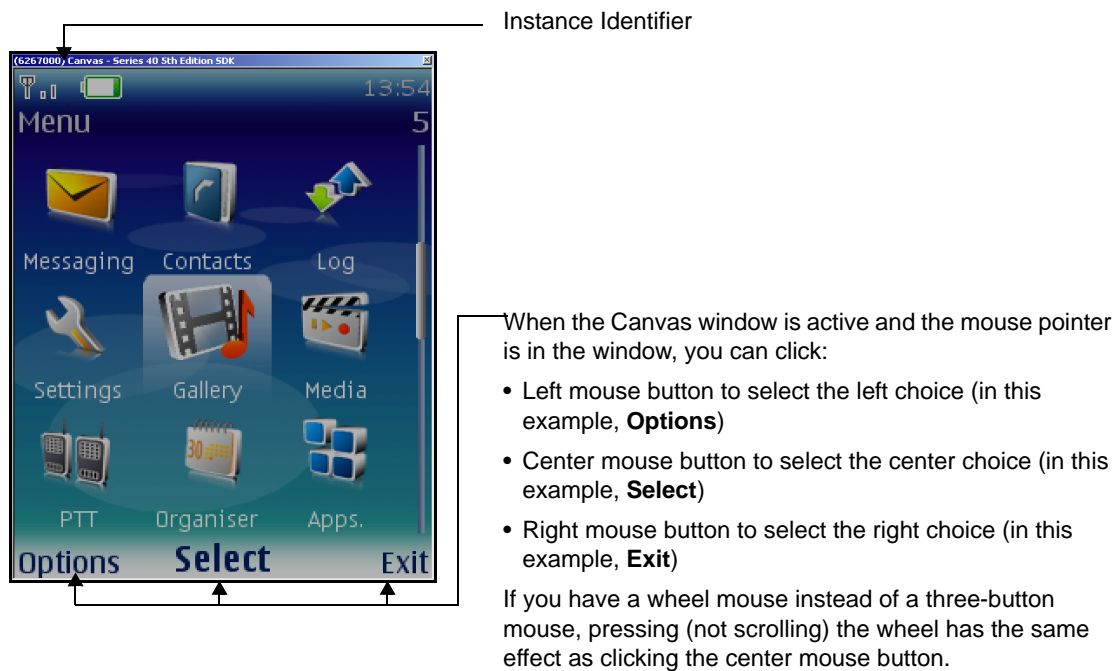
## Showing a magnified view of the main window

Content appears in the display area of the SDK's main window, which is 240 by 320 pixels by default. This default option does not provide pixel-to-pixel accuracy. In some cases, you may want to see a magnified view of the window so you can examine it in more detail. Use the Canvas tool supplied with the SDK to magnify the Canvas window.

Select **Tools>Canvas** and *one* of the following:

- 200% (480x640 pixels)
- 300% (720x960 pixels)

The Canvas window shows a magnified view of the LCD part of the main window:



In the Canvas window, you can:

- Select options by clicking the left, center (or wheel), or right mouse button to choose the corresponding choice displayed at the bottom of the window.
- Scroll with the mouse's scroll wheel.
- Type text directly into fields.

The display in the Canvas and the phone image are synchronized. They always display the same content.

Changing the display size of either the main window or the Canvas window does not affect the MIDP canvas dimensions for a MIDlet running inside the SDK. They are always the same as those in an actual handset.

## Inputting text and commands from a keyboard

In addition to clicking on the SDK display, you can use the computer keyboard to activate the SDK's soft keys and enter information into the SDK.

### Emulating mouse clicks with keyboard shortcuts

Here's a list of each keyboard key or key combination that performs an action on the SDK equivalent to clicking on the SDK display:

SDK Soft Key	Computer Keyboard Equivalent
Left softkey	F9
Middle softkey (Selection key)	F11 Enter
Right softkey	F12 Backspace
Arrow up key	Up arrow
Arrow down key	Down arrow
Arrow left key	Left arrow
Arrow right key	Right arrow
Send key (Call creation key)	NumPad/(slash)
End key (Call termination key)	NumPad - (minus)
Hash key (number sign)	NumPad . (period)
* (Star key)	NumPad *
Numbe keys 0 through 9	NumPad 0 through NumPad 9

### Entering text from the keyboard

You can type on your keyboard and have the characters appear in the display of the SDK. For example, if the SDK browser is currently displaying a text entry box contained in an XHTML page to which you browsed, you can enter characters into the box by typing on the keyboard. Characters entered into the SDK in this manner are not mapped directly to the SDK soft keys like the shortcut keys listed in [Emulating mouse clicks with keyboard shortcuts](#) on page 37.

You can use all the alphanumeric computer keys (a-z, A-Z, 0-9), some of the non-alphanumeric characters (!@#\$%^&\* ()+,: ' " , <.>/?[ ] \_ = { } | \ ~), and space key to enter text. Those characters are mapped to the corresponding non-numpad computer keyboard keys. This feature is not available for languages that are not based on a Latin alphabet, such as Arabic, Chinese (Simplified and Traditional), Hebrew, Thai, and Vietnamese.

The full keyboard input is implemented for text entry boxes only. You can use the numeric keypad to enter numbers. See [Using the numeric keypad](#) on page 22.

## Button chording

You can use your keyboard to enter multiple, simultaneous button presses (button “chording”). In a game MIDlet, for example, if the user presses and holds the up arrow button, the game target will move up. In the Nokia 6267 handset, if the user continues to hold the up arrow button and also presses and holds the right arrow button, the game target will move diagonally up and to the right.

In the SDK, the PC keyboard limitations might cause the MIDlet to behave differently. If you press and hold only one PC keyboard button, the target in the SDK behaves in the same way as in the Nokia 6267 handset. But if you press and hold, for example, two keyboard buttons at the same time, the target might behave differently, depending on the PC keyboard and the MIDlet in question.

You can use the mouse to click one key in the SDK handset, in addition to the key you are pressing on your PC keyboard.

## Working with content types

In the SDK, you can browse directly to some content types using **File>Open**. You can access other content types using a WML, HTML, or XHTML file.

### What content types are mapped to what file types in the SDK?

Because local files do not go through a server, the SDK provides an internal mapping between file types and content types. For example, when you use **File>Open** to load a local .jad file, the SDK uses content type `text/vnd.sun.j2me.app-descriptor`. The following table lists the mappings between file and content types:

File suffix	Content type presented to the SDK
.3g2	video/3gpp2 and audio/3gpp2
.3gp	video/3gpp and audio/3gpp
.3gpp	video/3gpp and audio/3gpp2
.amr	audio/amr

File suffix	Content type presented to the SDK
.3gs	video/3gpp2 and audio/3gpp2
.3gp	video/3gpp and audio/3gpp
.3gpp	video/3gpp and audio/3gpp2
.amr	audio/amr
.awb	audio/amr-wb
.bmp	image/bmp
.cod	text/x-co-desc
.dd	application/vnd.oma.dd+xml
.dm	application/vnd.oma.drm.message
.dr	application/vnd.oma.drm.rights+xml
.dcf, .df	application/vnd.oma.drm.content
.drc	application/vnd.oma.drm.rights+wbxml
.gif	image/gif
.html	text/html
.jad	text/vnd.sun.j2me.app-descriptor
.jar	application/java-archive
.jpg	image/jpeg
.jpeg	
.m3g	application/m3g
.m4v	video/mp4
.mid	audio/midi
.midi	
.mms	application/vnd.wap.mms-message
.mp3	audio/mpeg3
.mp4	video/mpeg
.mpeg	
.nrt	application/vnd.nokia.ringing-tone
.nth	application/vnd.nok-s40theme
.nwp	image/vnd.nok-wallpaper
.oplc	image/vnd.nok-oplogo-color
.ott	audio/vnd.nok-ringington
.png	image/png
.sic	application/vnd.wap.sic

File suffix	Content type presented to the SDK
.slc	application/vnd.wap.slc
.svg	image/svg+xml
.svgz	
.swf	application/x-shockwave-flash
.spmidi	audio/sp-midi
.smf	
.xmf	
.mxmf	
.text	text/plain
.txt	
.vcf	text/x-vCard
.vcs	text/x-vCalendar
.wav	audio/wav audio/mp3
.wbmp	image/vnd.wap.wbmp
.wml	application/vnd.wap.wml
.wmlc	application/vnd.wap.wmlc
.wmlsc	application/vnd.wap.wmlscriptc
.xhtml	application/xhtml+xml

## What content types can you load directly using the browser?

You can use the browser to load all of the content types mentioned in the previous section by:

- Making an HTTP request directly through **File>Open URL**. For example:  
`http://www.nokia.com`
- Using a hyperlink reference (HREF) within WML, HTML, or XHTML code to provide a link to another item, such as a new web page or image. For example:  
`<a href=http://www.nokia.com/Click_me.wbmp>Click me</a>`

In addition, the browser can load some additional content types:

- `application/x-nokiagamedata`
- `application/x-wap-prov.browser-bookmarks`
- `application/x-wap-prov.browser-settings`

You can reference WMLScript with a hyperlink only within WML, HTML, or XHTML content. The browser cannot directly open these formats:

- `application/vnd.wap.wmlsc`



In-line content types within WML, HTML, and XHTML code display as a subcomponent of a page. Generally, all content types can be referenced, but only images can be displayed in-line. However, some exceptions apply. For example, wallpaper (image/vnd.nok-wallpaper) can be displayed in-line but cannot be referenced and a snapshot (image/vnd-nok-camera-snap) can be referenced but cannot be displayed in-line.

An example of an in-line format is:

```

```

The content types you can use in-line are:

- image/gif
- image/jpeg, image/jpg
- image/png
- image/vnd.wap.wbmp
- image/bmp

## What audio content types does the SDK support?

You can play the following audio content types from files:

- application/vnd.nokia.ringing-tone
- audio/3gpp
- audio/3gpp2 (containing AMR or AMR-WB)
- audio/amr
- audio/awb
- audio/mid\*
- audio/midi\*
- audio/x-mid\*
- audio/x-midi\*
- audio/sp-midi\*
- audio/vnd.nokia.ringing-tone
- audio/vnd.nokia.mobile-xmf
- audio/vnd.nokia.mobile-xmf
- audio/wav
- audio/mp3

\*With up to 64 channels

The SDK simulates playing audio/3gpp and audio/3gpp2 (containing AMR or AMR-WB).

You may play these content types in the handset but not in the SDK:

- audio/3gpp2 (containing AAC)
- audio/aac
- audio/mpeg3

- audio/mpeg4

If you attempt to play one of these content types in the SDK, the SDK sends an error message to the Diagnostics log.

The handset supports audio capture from its microphone and the FM radio and stores it as audio/amr content. The SDK does not support a microphone. However, it simulates recording from a microphone using pre-recorded content installed with the SDK. The SDK does not support recording from the FM radio.

If the content is in stereo, the SDK plays it in stereo on the computer's sound card. However, the SDK does not emulate the enhanced stereo separation supported in the handset.

## What video content types does the SDK support?

You can play the following video content types from files:

- video/3gpp
- video/3gpp2 (h263 frames only)
- video/mpeg (containing 3GPP, 3GPP2, MPEG1, or MPEG2)

You can play the following content types in the handset but not in the SDK:

- video/mp4
- video/mpeg (containing MPEG4)

If you attempt to play one of these content types in the SDK, the SDK sends an error message to the Diagnostics log.

The SDK simulates playing video, but does not show the video image in the SDK LCD display area. Instead, the SDK displays a reference video image, while the video is playing.

The handset supports capturing still images in the image/jpeg format and video clips in the video/3gpp format using its camera. The SDK does not support a camera; however, it simulates still image capture using pre-recorded content installed with the SDK. The SDK does not simulate video clip capture.

## Generating events

Generated events let you evaluate what happens to the currently running application when a typically unplanned event occurs, such as an incoming call or a lost network signal.

To generate one of these circumstances, select **Tools>Generate Event** and *one* of the following options:

- **Incoming call** - Simulates an incoming call. This event does not play ring tones or vibrate the phone image to behave like a real handset.

- **No signal** - Simulates a handset that loses the signal from a GSM/GPRS network. This state remains until you turn it off.

## What's next?

Now that you've loaded and viewed content on the SDK, you can get information about that content in [Getting Information About the SDK's Content](#) on page 45.

Or, you can begin to develop:

- Messages and browser content. See [Using the SDK for Message Content Development](#) on page 65.
- MIDlets. See [Using the SDK for MIDlet Development](#) on page 77.

You might also want to explore running the SDK as a standalone application from a command line. See [Running the SDK from a Command Line](#) on page 85.



---

# 4

## Getting Information About the SDK's Content

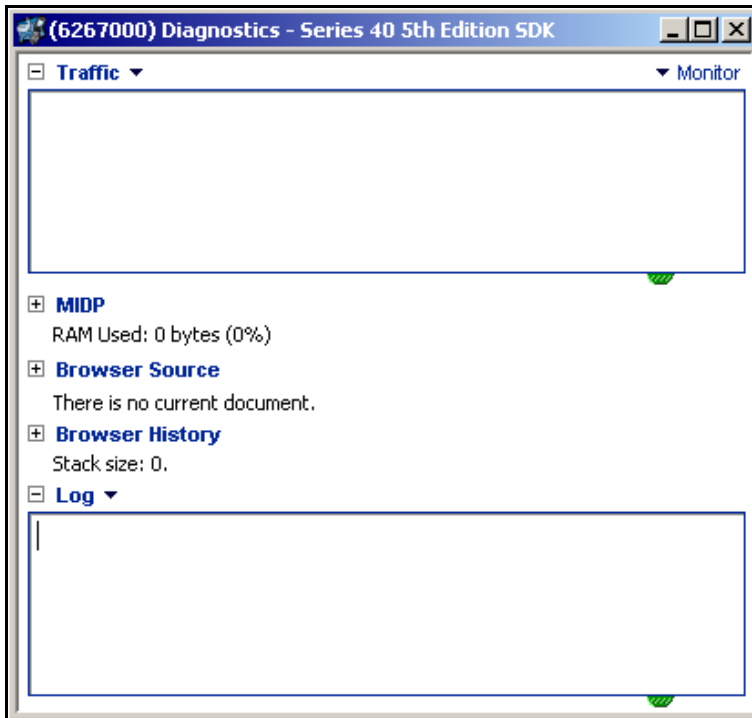
To help you analyze and troubleshoot the content on the SDK, you can view information about the current content on the SDK in the Diagnostics window. The information in this window is dynamic and changes as the content on the SDK changes.

If you do not need the information in the Diagnostics window and want to make the SDK start faster, you can disable the window. You might want to turn off the Diagnostics window, if you are running the SDK through an IDE for MIDlet development and are running tracing through the IDE. See [Setting the general behaviors of the SDK](#) on page 20.

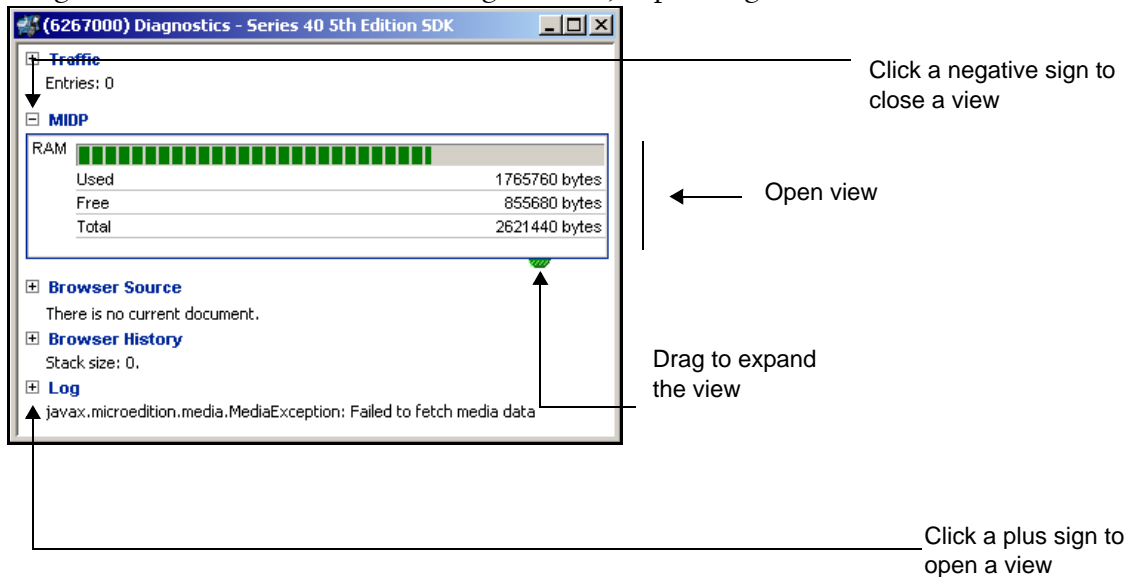
## Working with the Diagnostics window

The Diagnostics window displays information that can help you determine whether the content on the SDK is working properly. To view the Diagnostics window, make sure the Diagnostics windows is enabled (see [Setting the general behaviors of the SDK](#) on page 20) and select **Tools>Diagnostics**.

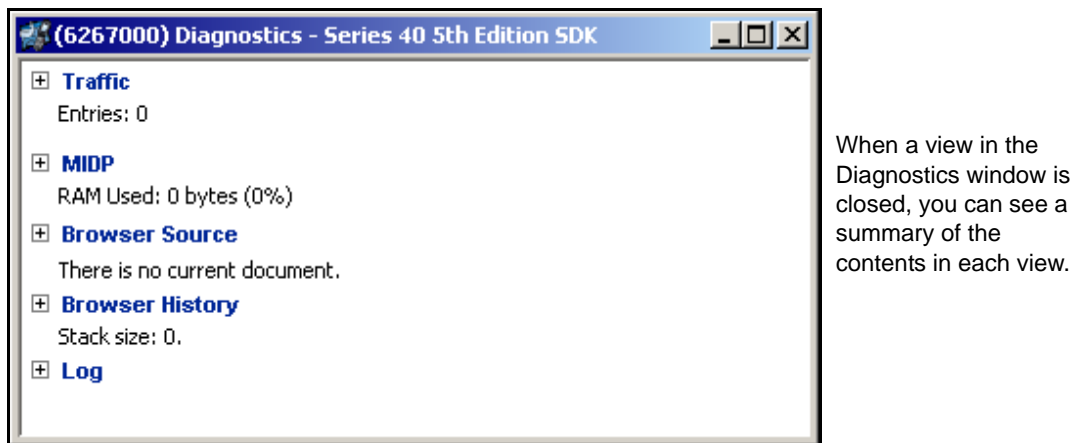
If nothing is loaded on the browser, the Diagnostics window looks like this when you first open it:



You can open, close, and adjust the size of a view. With an open MIDP view, the Diagnostics window looks something like this, depending on what's loaded on the SDK:



When content is loaded on the SDK and none of the views are open, the Diagnostics window looks something like this, depending on the content:



To display only the views:

- 1 Right-click in the Diagnostics window and select **Views** from the menu that appears. A list of views appears.
- 2 Uncheck the views you do not want to see. The checked views expand to fill the space of the Diagnostics window.

The table below lists what information can be viewed from the Diagnostics window's different views:

To view	See
Information about requests to and responses from the Internet	<a href="#">About the Traffic view</a> on page 48
Information about KVM status	<a href="#">About the MIDP view</a> on page 51
Source code of the content that is currently loaded in the SDK browser	<a href="#">About the Browser Source view</a> on page 52
A history list of the SDK browser	<a href="#">About the Browser History view</a> on page 56
Warning messages	<a href="#">About the Log view</a> on page 57

## About the Traffic view

The Traffic view displays events that occur between the SDK and an external entity, such as a gateway simulator or the Internet. (To trace network events that occur in the KVM, see [Setting MIDP tracing and speed options](#) on page 28.)

The events are triggered as the content on the SDK changes:

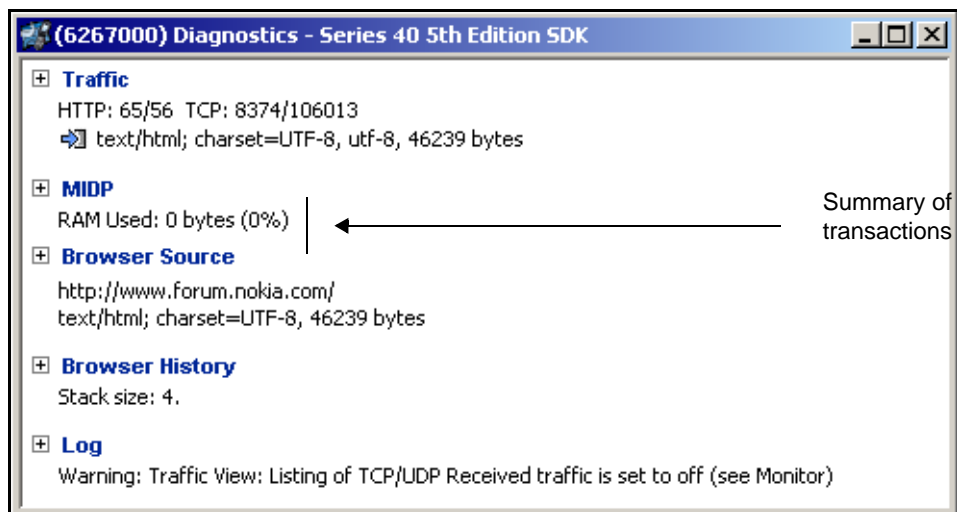
This event	Is triggered by
HTTP or HTTPS requests and responses	A MIDlet, the browser, or any other built-in phone application sends an HTTP/HTTPS request or receives a response to such a request.
TCP stream data sending and receiving	A MIDlet, the browser, or any other built-in phone application sends or receives TCP data.
UDP datagram sending and receiving	A MIDlet or built-in phone application sends or receives UDP data.
MMS message sending and receiving	A MIDlet or built-in phone application sends or receives an MMS message. This includes: <ul style="list-style-type: none"><li>• An SDK instance sends or receives an MMS from another SDK instance.</li><li>• The user opens an MMS message using <b>File&gt;Open</b>.</li><li>• An external tool (such as NMIT) pushes an MMS message to the SDK.</li></ul>
WAP push message receiving	The SDK receives a WAP push message either from <b>File&gt;Open</b> or from an external tool (such as NMIT).



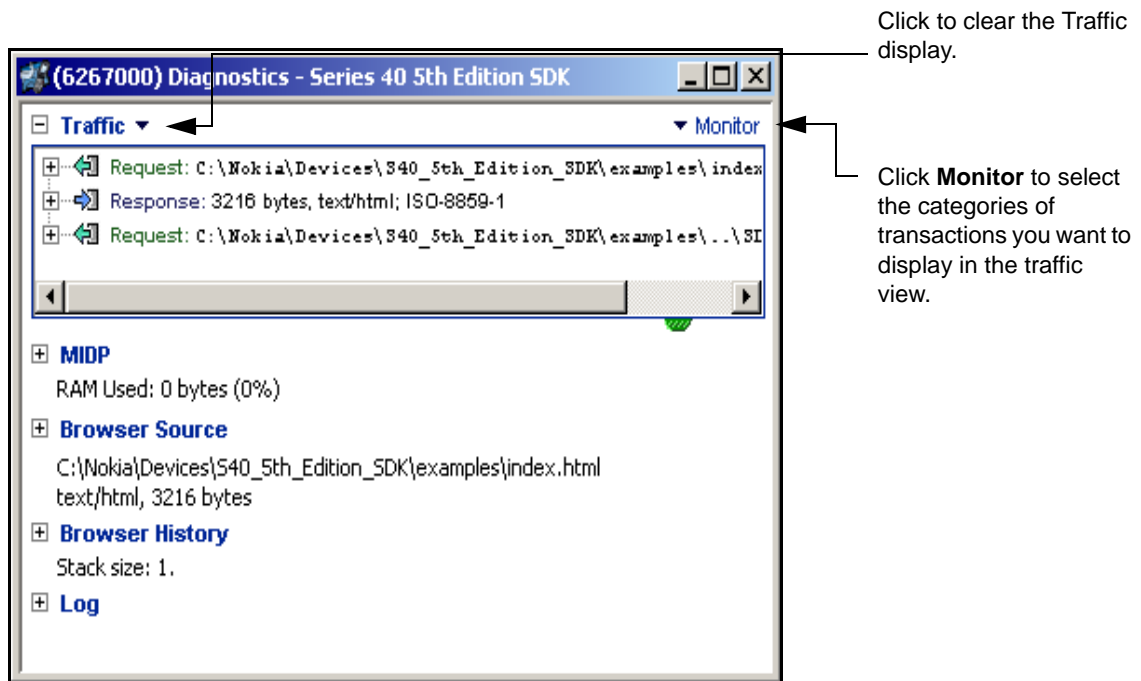
This event	Is triggered by
SMS message fragment sending or receiving	A MIDlet or built-in phone application sends or receives an SMS message fragment.
SMS message sending or receiving	A MIDlet or built-in phone application sends or receives an SMS message.
Bluetooth data sending or receiving	A MIDlet or built-in phone application sends or receives data via Bluetooth connectivity. This connectivity is simulated in the SDK, by using the Nokia Connectivity Framework.

When the Traffic view closes, this summary displays:

- Number of HTTP requests and responses.
- Number of bytes sent and received on the TCP layer.
- Number of bytes sent and received on the UDP layer.
- Number of SMS fragments (PDUs) sent and received.
- Number of bytes sent and received via Bluetooth.



When you open the Traffic view, you see the current list of transactions that have occurred, while the SDK is communicating:



The most recent transaction appears at the bottom of the list.

To display specific transaction types, click **Monitor** and check the transaction categories you want to display. The information about the transactions you don't check is available but does not display until you check the transaction. These settings are persistent between SDK instance invocations. The categories are:

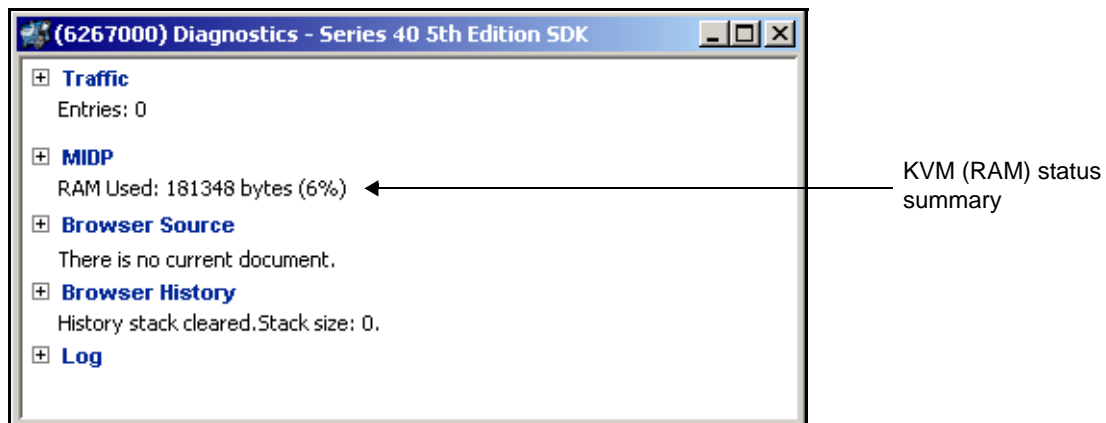
Check this	To display information about
Requests	Requests that originate in the SDK. For example, when the SDK sends a request to a web server, the transaction is recorded under Requests.
Content Responses	Responses made to the SDK as a result of a request that the SDK made. For example, a website responds to the SDK's request for the site by loading it on the browser. The transaction is recorded under Content Responses.
Pushed Messages	A transaction that is not a result of a request the SDK made. Examples of these transactions include Push and MMS messages that you can send to the SDK from IDEs, the CLI, or <b>File&gt;Open</b> .
SMS Sent	SMS messages sent, including the size of whole messages and message fragments.

Check this	To display information about
SMS Received	SMS messages received, including the size of whole messages and message fragments.
TCP/UDP Sent	Low-level protocol messages sent.
TCP/UDP Received	Low-level protocol messages received.
Bluetooth Sent	The size of Bluetooth data sent via the simulated Bluetooth connectivity.
Bluetooth Received	The size of Bluetooth data received via the simulated Bluetooth connectivity.

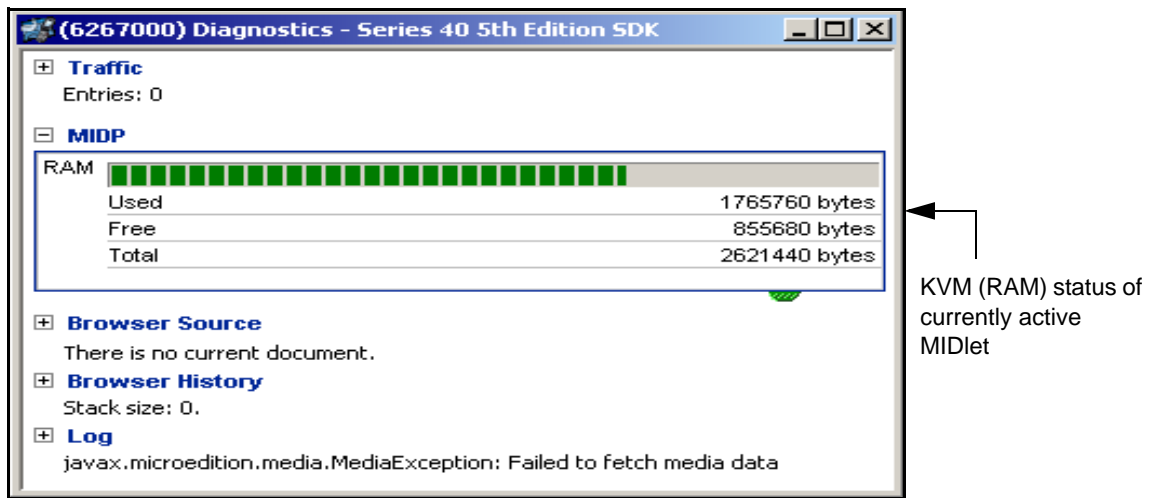
## About the MIDP view

The MIDP view provides information about the MIDP Java KVM (kilobyte virtual machine). The size of the KVM memory heap (heapsize) controls how much memory a MIDlet can take to run on the SDK. (See [About KVM heapsize status](#) on page 52.) The KVM status is updated as a MIDlet runs.

When the MIDP view closes, the Diagnostics window looks like this:



When you open the MIDP view, you see the current status of the KVM (RAM) storage:



## About KVM heapsize status

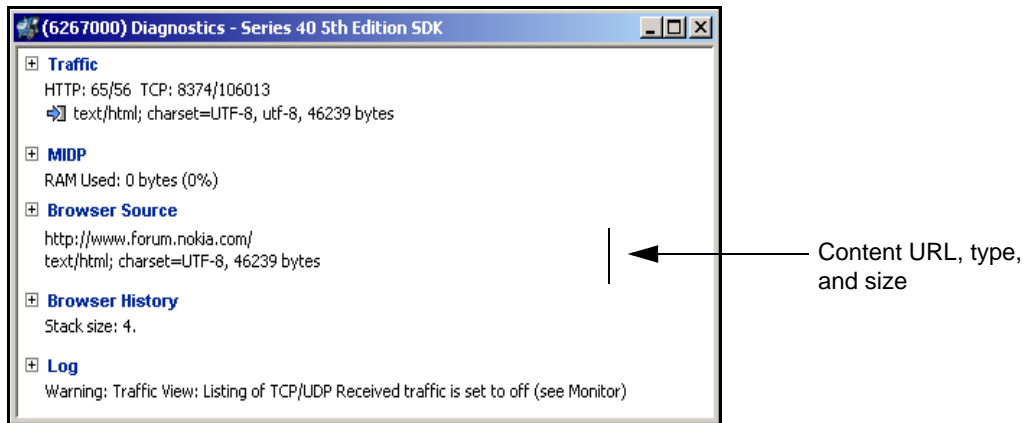
For the MIDlet that is currently active on the SDK, the KVM memory status shows in bytes:

- **Used:** Bytes of the KVM heap that are in current use.
- **Free:** Bytes in the KVM heap that are available for use.
- **Total:** The total size of the KVM heapsize. (To modify the heapsize, see [Setting the KVM heapsize](#) on page 27.)

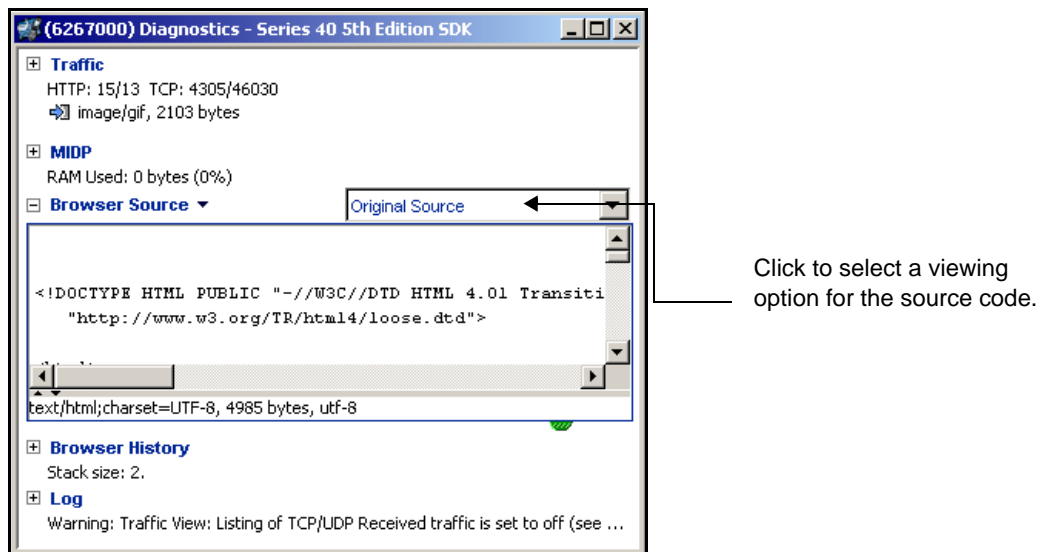
## About the Browser Source view

The Browser Source view displays the source code of the content that is currently loaded on the SDK browser.

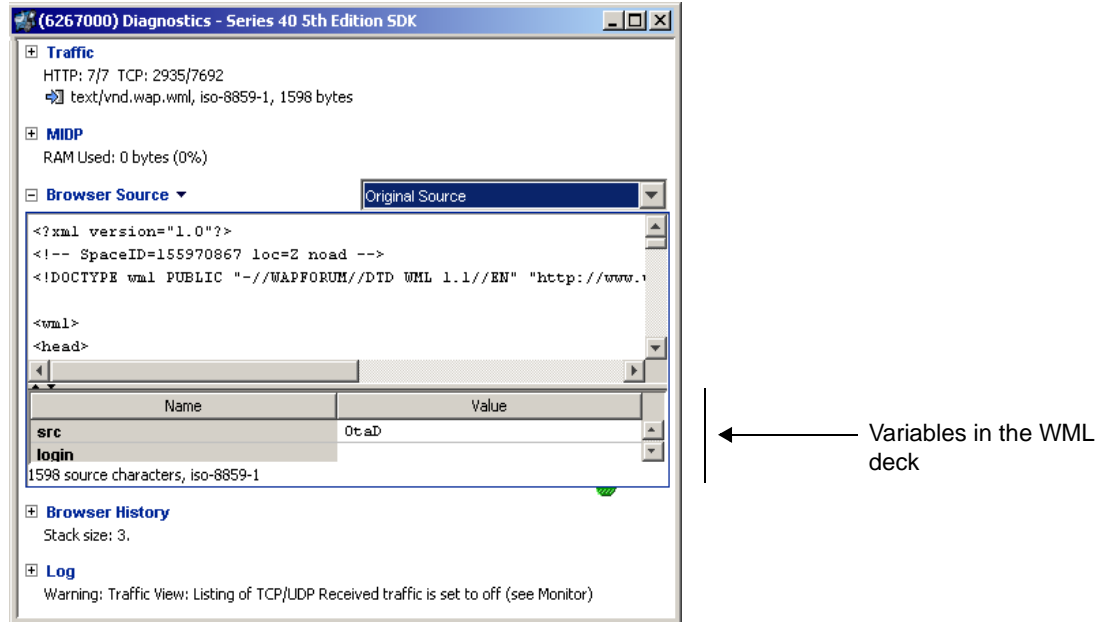
When the view is closed, you see the URL of the current content, its file type, and size.



When you open the Browser Source view, you see the source code of the content that is on the browser:



When the browser displays WML content that contains a deck with variables, an additional section displays the variables and their values:

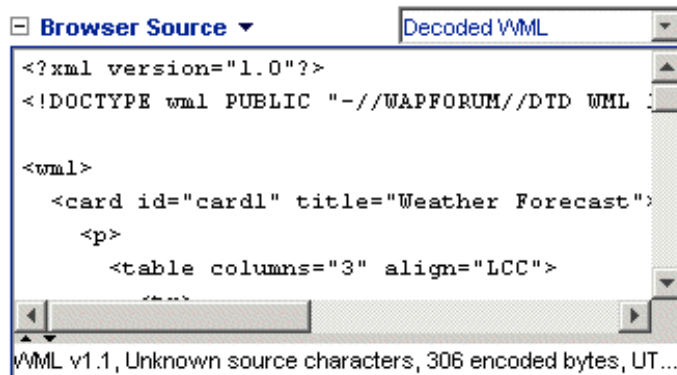


You can view the current content in several formats, depending on the content that's on the browser. To see what views are available, click on the drop-down menu on the top-right corner of the Browser Source view. The following options are available under these circumstances:

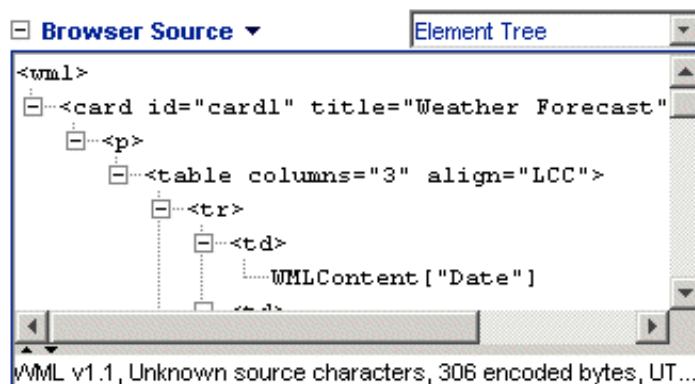
If the current content is	You can view the source as
Unencoded, as in XHTML or WML	Original source code
Encoded, as in binary WML (.wmlc)	Decoded WML
	Element Tree
	Byte Code

When the content is encoded WML, you can choose from the following three views:

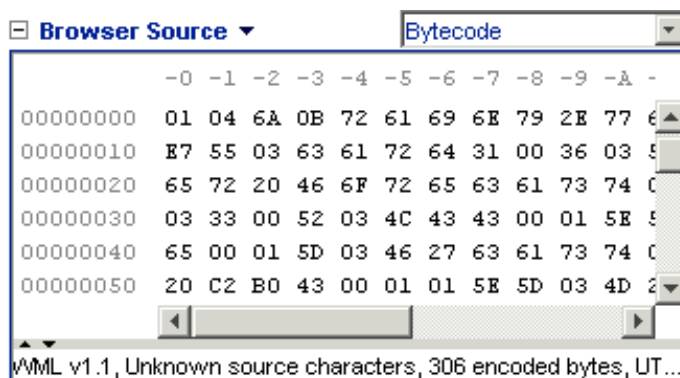
- The **Decoded WML** view displays the result of decoding an encoded response received from a server.



- The **Element Tree** view displays a parse tree of the current content as the SDK received it. This view is useful for detecting the hierarchical structure of the content.



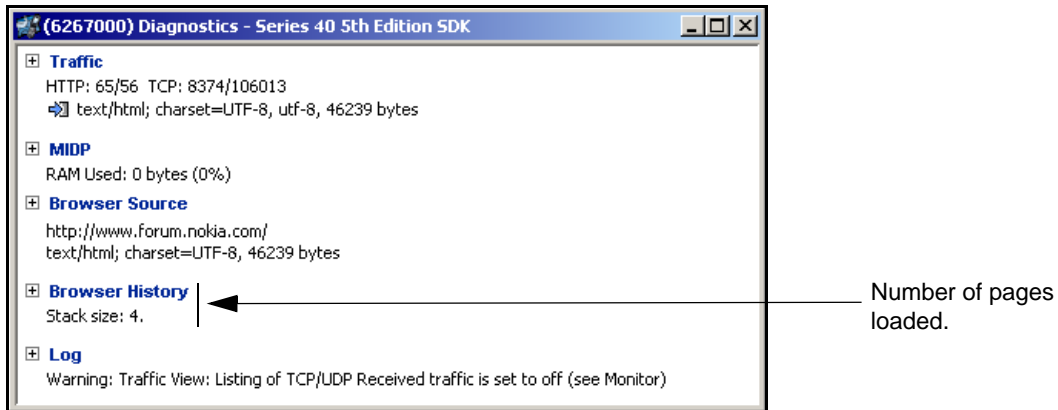
- The **Bytecode** view displays the bytes the SDK receives. Some content types, such as images, are available only in bytecode.



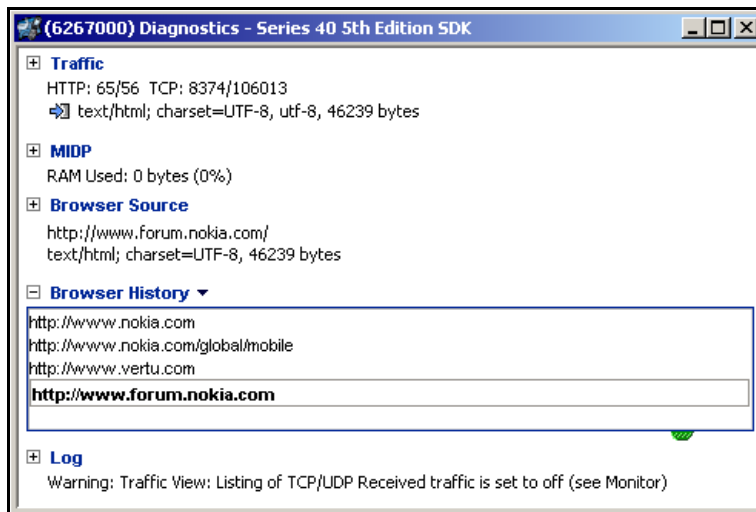
## About the Browser History view

The Browser History view displays a list of sites that appeared on the browser during the current session.

When the Browser History view closes, you see a summary of the number of sites listed in the open view:



When you open the view, you see the list of locations with the most recent location at the bottom. The location that currently displays on the browser is highlighted:



The Browser history on the SDK is erased when you:

- Quit the browser.
- Open local content or browse to content using **File>Open** or **File>Open URL**.



To preserve the browser history when you change websites, change websites as you would on the handset. For example, the following example shows how to add bookmarks in the SDK's handset menu and how to preserve the browser history while using the bookmarks for browsing:

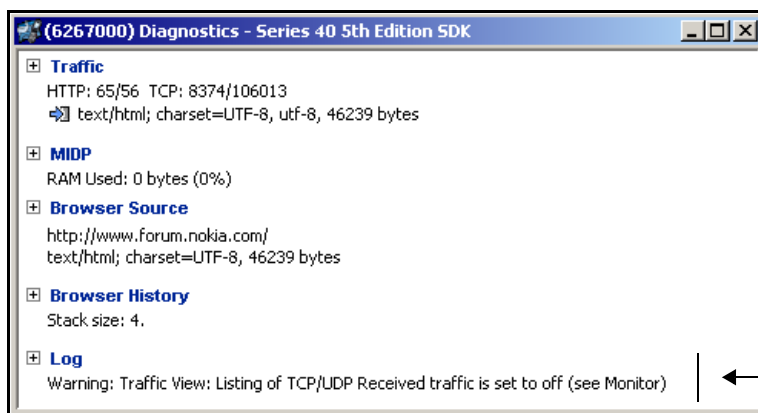
- 1 Go to the idle screen by clicking the **SDK End Call** soft key one or more times. You know you are on the idle screen when you see **Go to**, **Menu**, and **Names** on the display.
- 2 Perform the following actions:
  - Select **Menu>Web>Bookmarks>Add**. Enter `http://www.vertu.com` and select **OK**. Enter Vertu for the title and select **OK**.
  - Select **Options>New bookmark**. Enter `http://www.n-gage.com` and select **OK**. Enter N-gage for the title and select **OK**.
  - Select **N-gage**. The `www.n-gage.com` web page loads on the browser, and the browser history records the URL.
  - Select **Options>Bookmarks>Vertu**. The `vertu.com` web page loads on the browser, and the browser history records the URL without deleting the N-gage listing.

## About the Log view

The Log view displays messages that the SDK generates, such as warnings about unsupported content types and file sizes that exceed what the SDK can handle. The Log view also informs you about which monitors are turned on.

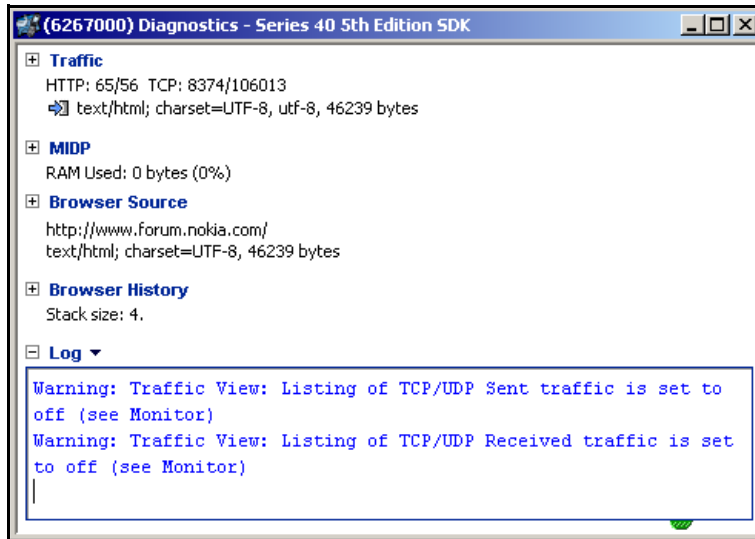
Smart card reader warnings...

When the Log view closes, the Diagnostics window looks like this:



← Last message reported in the log.

When you open the Log view, you see any messages that the SDK generates:



## Working with the PC file system

The SDK duplicates part of the phone's file system on your computer. This allows you to preinstall content into the SDK when you start it and to observe how the SDK changes files.

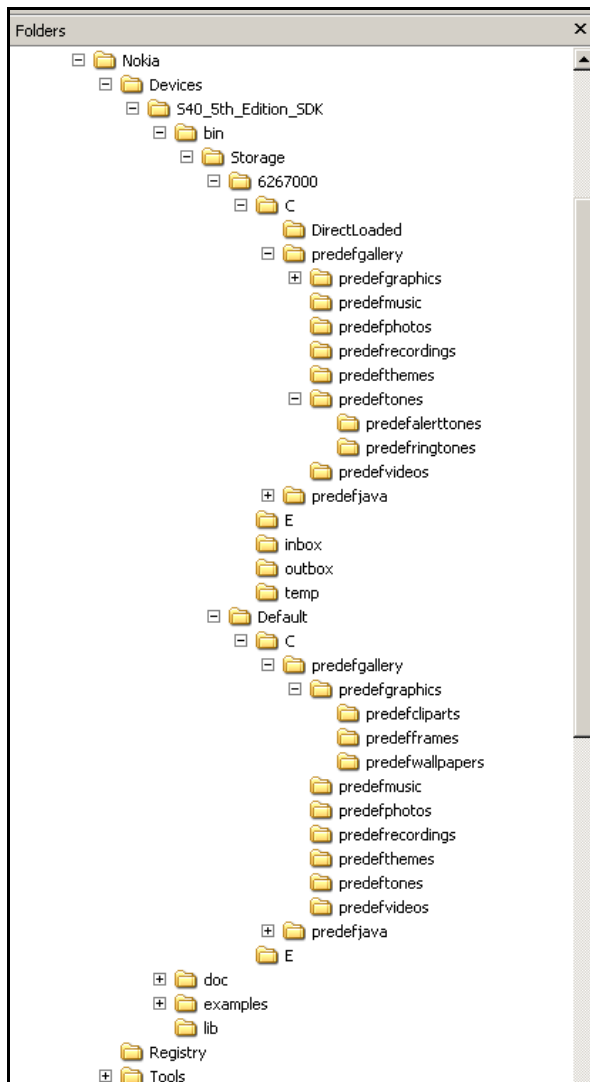
The first time you create a new SDK instance, the SDK creates a directory for that instance (the instance directory) at:

`<SDK installation directory>\bin\Storage\<instance identifier>`

by copying the contents of the `Default` directory to the new location. Any files located in the `Default` directory are duplicated in the instance directory.

As the SDK starts, it copies any files in the instance directory into the phone's file system. After the SDK is running, you can use the SDK display to access these files in the phone's file system.

For example, when you create an SDK instance with an identifier of 6267000, the directory structure looks like this:



Within the instance directory, the C directory corresponds to the phone's file system and the E directory corresponds to its memory card. Within the C directory:

- The `predefgallery` directory corresponds to the part of the phone's file system used to store audio and image files. To view these files, select **Menu>Gallery** in the SDK display.
- The `predefjava` directory corresponds to the directory used to store Java applications. To view these files, select **Menu>Applications** in the SDK display.
- The `DirectLoaded` directory does not correspond to any part of the phone's file system. The SDK uses this directory for temporary storage of Java MIDlets that you run using the Direct Loading method. See [About OTA provisioning and direct loading of a MIDlet](#) on page 78.

After you create an SDK instance, you can modify the contents of the instance directory using the SDK window or the Windows tools on the computer.

- If you modify the content of the phone's file system using the SDK display, the SDK immediately duplicates the modification in the instance directory on your computer.
- If you modify the files in the instance directory using the Windows tools, the modifications only become visible in the SDK display the next time you start that SDK instance. The SDK instance reads the contents of the instance directory only at startup.

The `E` directory corresponds to the part of the phone's file system that accesses the memory card. To view these files, select **Menu>Gallery>Memory card** in the SDK display. If you simulate removing the memory card (see [Emulating a memory card](#) on page 21), the SDK does not remove the files in the `E` directory on your computer, but they are no longer accessible to the SDK instance.

The contents of an instance directory are accessible only to an SDK instance with a matching instance identifier. If you start an SDK instance using an identifier that corresponds to an existing instance directory, the SDK instance uses that existing directory.

You can modify the `Default` directory using the Windows tools on the computer but not using the SDK display. For example, to add an image for use in all future SDK instances, copy the image into the `Default\predefgallery\predefphotos` directory with a Windows Copy and Paste.

If you modify the `Default` directory, those modifications are copied to a new instance directory the first time you create a new instance. However, those modifications are not copied to any existing instance directories.

## What's next?

You can:

- Set up SDKs to send and receive SMS, as described in [Simulating Wireless Connectivity between SDKs](#) on page 61.
- Develop messages, as described in [Using the SDK for Message Content Development](#) on page 65.
- Develop MIDlets, as described in [Using the SDK for MIDlet Development](#) on page 77.

---

# 5

## Simulating Wireless Connectivity between SDKs

The Series 40 5th Edition SDK simulates wireless connectivity technology between SDKs with NCF Lite 1.2, a supporting application that enables SMS and Bluetooth connectivity between SDKs. The simulation runs over a standard personal computer network using the TCP/IP protocol and does not use hardware for wireless communications.

NCF Lite 1.2 is automatically installed when you install the Series 40 5th Edition SDK.

The Series 40 5th Edition SDK supports simulated SMS in MIDlets using an implementation of JSR 205, which includes JSR 120. The SDK also supports standard textual SMS messages, and Bluetooth connectivity between Series 40 5th Edition SDK instances, through the phone menus shown in the display area.

## Working with the wireless simulation

You can connect a maximum of four SDKs at one time when you work with the wireless connectivity simulation. The SDKs can be:

- Two SDK instances that conform to the Series 40 Developer Platform 2.0 or later. You can do this with NCF Lite 1.2.
- A Series 40 5th Edition SDK instance and an S60 SDK instance.  
To connect a Series 40 and an S60 SDK instance, you'll first need to download the full version of NCF. This version is freely available from [www.forum.nokia.com](http://www.forum.nokia.com).

For information about connecting a Series 40 SDK and a S60 SDK, see the NCF documentation that comes with the download. Installing the NCF version with full features automatically replaces NCF Lite 1.2.

### Starting NCF

You must start NCF Lite 1.2 or the full version of NCF to enable transactions between two SDKs for SMS, Bluetooth and MIDlets that use SMS (JSR 205 or JSR 120).

To start NCF Lite 1.2:

- 1 Select **Start>Programs>Nokia Developer Tools>Nokia Connectivity Framework>Nokia Connectivity Framework Lite**.

NCF Lite 1.2 begins working in the background and displays this icon on the system tray.




To start the full version of NCF:

- 1 Select **Start>Programs>Nokia Developer Tools>Nokia Connectivity Framework>Nokia Connectivity Framework Full**.

Depending on the speed of the computer you are using, you might need to wait between 10 and 40 seconds for NCF and all its components to become available to the SDK. If you start the SDK before the NCF components are available, you might get a message that the SDK does not have a MAC address or the SDK might not work properly. If this happens, shut down the SDK and NCF (in that order). Restart NCF and the SDK (in that order), leaving sufficient time between the two applications for NCF to become fully functional.

### Shutting down NCF Lite 1.2

Exit from the SDK before you shut down NCF Lite 1.2.

To turn off NCF Lite 1.2, click the  icon in the system tray and select **Shut Down**.

## What's next?

You might want to explore developing messages, as described in [Using the SDK for Message Content Development](#) on page 65.





---

# 6

## Using the SDK for Message Content Development

You can create MMS and DRM messages or content on an application and send them to the SDK within an IDE. You can also load a UI theme (.nth file) on the SDK from a local file or from a website.

If you are planning to send SMS messages directly between SDKs instead of pushing them from an IDE, see [Simulating Wireless Connectivity between SDKs](#) on page 61.

## Working with MMS messages

The SDK supports MMS messaging using the simulated handset keypad, NMIT and the JSR 205 MMS API. An MMS message contains rich media content that may include one or more text, image, or audio components. Some content types can only be received; they cannot be forwarded. For example, you can't forward a MIDI content type (`audio/midi`) that contains a ring tone you've purchased. See [Creating and sending an MMS message directly on the SDK](#) on page 67.

The SDK can handle an MMS message whose content contains one or more parts of the following content types:

Content type	File format	Can only be received?
<code>image/jpeg</code>	JPEG	
<code>image/gif</code>	GIF87a, GIF89a	
<code>image/png</code>	PNG	
<code>image/bmp</code>	BMP	
<code>image/vnd.wap.wbmp</code>	WBMP	
<code>image/vnd.nok-wallpaper</code>	wallpaper	Yes
<code>audio/amr</code>	voice clip	
<code>audio/midi</code> <code>audio/mid</code> <code>audio/x-mid</code> <code>audio/sp-midi</code> <code>audio/x-midi</code>	MIDI	Yes
<code>text/plain</code>	text	
<code>application/vnd.oma.drm.message</code>	Digital rights message - Forward Lock	Yes
<code>application/vnd.oma.drm.message</code>	Digital rights message - Combined Delivery	Yes
<code>application/vnd.oma.drm.content</code>	Digital rights content - Separate Delivery	
<code>application/vnd.oma.drm.rights+wbxml</code>	Digital rights - Separate Delivery	Yes
<code>video/3gpp</code>	Video clip	

Content type	File format	Can only be received?
text/x-vCalendar	Calendar note	
text/x-vCard	Business card	

## Creating and sending an MMS message directly on the SDK

Two or more SDKs can exchange MMS messages using the `-Xinbox` and `-Xoutbox` command line options. You may want to perform this test to see whether a MIDI file can be forwarded in an MMS message or to see how an MMS message appears on this SDK and future SDKs.

By default, each SDK instance creates both an MMS outbox and inbox folder on your computer at:

`<SDK installation directory>\bin\Storage\<instance identifier>`

The SDK places sent MMS message files into the outbox folder. It also monitors its inbox folder. If an MMS message file appears in the inbox folder, the SDK reads the file, treats it as a received MMS message, and deletes the file.

If you create two SDK instances, you can copy or move an MMS message file from the outbox folder of one instance to the inbox folder of the other, which allows you to simulate sending an MMS message from one instance to the other.

To set up a situation where two SDKs share their inboxes and outboxes:

- 1 Enter the following command to create an SDK instance that automatically writes message files to `box_a` and reads message files from `box_b`:

```
emulator -Xoutbox:c:\box_a -Xinbox:c:\box_b
```

- 2 Open a second command window. Enter the following command to create an SDK instance that automatically reads message files from `box_a` and writes message files to `box_b`:

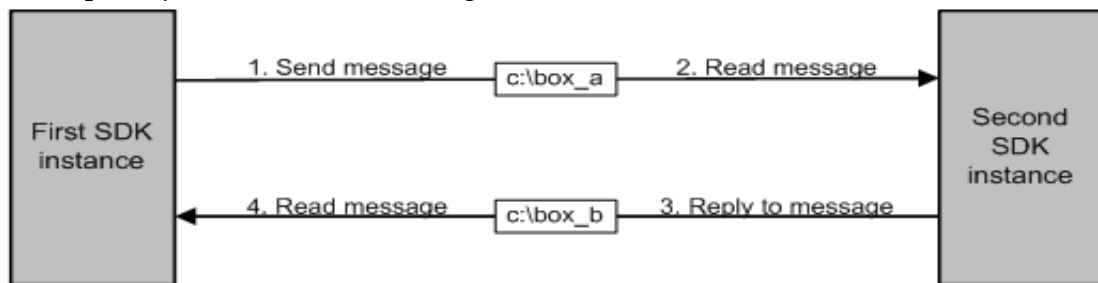
```
emulator -Xinbox:c:\box_a -Xoutbox:c:\box_b
```

- 3 Perform the following actions to create and send a message on the first SDK instance using the simulated phone menus:

- Select **Menu>Messaging** icon.
- Select **Create Message>Multimedia>Insert>Text** and type the text of the message. Select **Next**.
- Perform the appropriate action for graphics, video, and/or sound:
  - For graphics, select **Insert>Image>Open Images**. Choose the graphic you want and select **Insert**.
  - For video, select **Insert>Video clip>Open Gallery>Video clips**. Choose the video you want and select **Insert**.

- For sound, select **Insert>Sound clip>Open Gallery>Music files**. Choose the sound you want and select **Insert**.
  - Select **Options>Send** to set up how and to whom you want to send the message.
  - Select **Phone number** and enter any valid phone number. Optionally, enter a subject for the message.
  - Select **Options>Send**. The first SDK sends the message to `box_a`. The second SDK instance automatically retrieves the message file.
- 4 Read the message on the second SDK instance using the simulated phone menus and reply to or forward the message. The second SDK instance automatically sends the message to `box_b`. The first SDK instance automatically retrieves the message file.
  - 5 Read the forwarded message on the first SDK instance using the simulated phone menus.

Conceptually, the flow of the message looks like this:



**Note:** If a message cannot be forwarded because it contains a protected format, such as MIDI, the SDK displays an announcement stating that it cannot send the message.

You can create batch files to set up the exchange of MMS messages. The two batch files would look something like this:

```
@emulator -Xnew -Xoutbox:c:\box_a -Xinbox:c:\box_b
and
@emulator -Xnew -Xinbox:c:\box_a -Xoutbox:c:\box_b
```

You can take advantage of the `-Xinbox` option, if you set up a deployment directory in an IDE to automatically create MMS messages in the correct directory. Alternatively, you can manually place a message file into the deployment directory.

In either case, the SDK automatically retrieves the MMS message files from that directory, displays the message, and deletes the file.

## Creating and previewing an MMS message using NMIT

To create an MMS message using NMIT:

- 1 In NMIT, select **File>New>Messaging>MMS Wizard** to open the MMS Setup Wizard.
- 2 Within the wizard, specify the recipients of the message and the media files to be included in the message. The wizard supplies required headers with default header values that you can change in the following step. Click **Finish**. The MMS Message Editor window opens.
- 3 You can edit the header values for the entire MMS message and the individual parts. You can also add optional headers.
- 4 Select **Tools>SDK Control Panel** to display phone SDKs that NMIT detects.
- 5 To start a phone SDK, click the **Start** icon next to its name. NMIT starts the Series 40 5th Edition SDK and the panel displays the SDK instance identifier.
- 6 Click **Push** within the MMS Message Editor to send the MMS message to the SDK.  
The SDK displays a message that an MMS message has arrived. However, if the SDK is in browsing mode, the MMS message announcement does not display, and you must use the phone menus to quit the browser to see the announcement.
- 7 To close an SDK instance, select **File>Exit** from the SDK, even if you have started the instance from within NMIT.

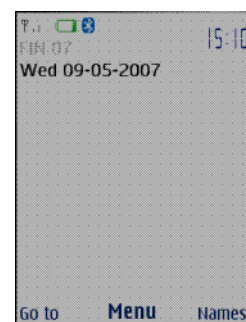
## Processing an MMS message on the SDK


This example shows you how to receive, view, and process an MMS message on the SDK. You must exit from the browser before you can send or receive an MMS message.

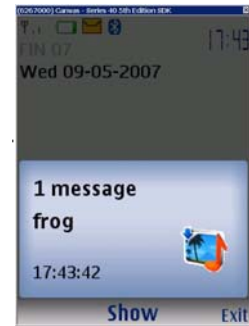
The MMS message in this example consists of two media parts, a `.txt` file and a `.jpg` image. The message also contains a `.smil` file that coordinates the timing of the display of the `.txt` and `.jpg` files.

To display the message-received announcement:

- 1 Go to the idle screen by clicking the SDK **End Call** soft key one or more times. You know you are on the idle screen when you see **Go to**, **Menu**, and **Names** on the display.



- 2 When the message arrives, the SDK displays a notification message as well as  (a message icon) at the top of the screen.



- 3 Select **Menu>Messaging>Inbox>Open** to display the message.



## Working with DRM messages

DRM is a system of content types and mobile phone behavior that allows a content provider to control how a phone user uses and re-distributes media content (such as a ring tone or a game). This control is achieved by pairing the media content with rights that control how often and for how long a phone user can use the content on the mobile phone. For example, a content provider can permit a phone user to receive and play a song once as a preview. Or, the content provider can allow the phone user to purchase the rights to play the song indefinitely.

DRM provides the following three message types that protect content with varying levels of security:

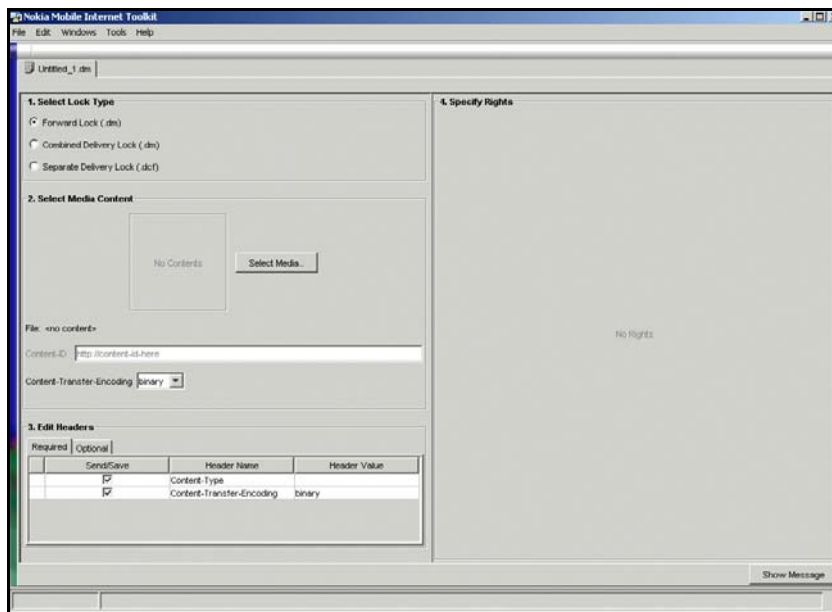
DRM message type	Description
Forward Lock	A single message that allows a phone user to use the content without limits on one device (mobile phone), but prohibits the user from forwarding the content to another device.
Combined Delivery Lock	A single message that allows a phone user to use the content with or without limitations on one device, but prohibits the phone user from forwarding the content to another device.

DRM message type	Description
Separate Delivery Lock	A pair of messages (one with content, one with the rights to the content) that allows a phone user to use the content with or without limitations on one device, if the user has received the rights to use the content. The phone user can forward the contents to another phone user but not the rights. Information about obtaining rights is embedded in the content file, so a new receiver of the content can obtain the rights to use the content.

## Creating a DRM message in NMIT

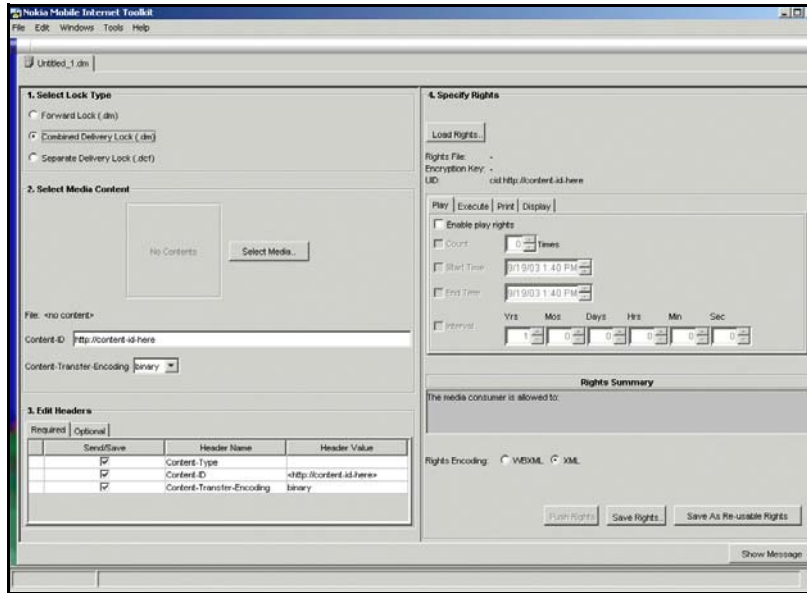
To create a DRM message using the **SDK control panel** in NMIT and send it to the SDK:

- 1 From the **SDK control panel** in NMIT, open the Series 40 5th Edition SDK, where you will later display the DRM message.
- 2 Select **File>New**. The **Available Content Type** window appears.
- 3 Click the **Deployment** tab to access the DRM editors.
- 4 From the **Deployment** tab, select the **DRM Message** icon, and click **OK**. The **DRM Message Editor** window appears:



The content of the DRM window varies, depending on the type of message you select. The image above shows the DRM window when you select **Forward Lock**.

When you select **Combined Delivery Lock** or **Separate Delivery Lock**, the digital rights side of the window appears:



In this side, you specify supported content (for all message types).

On this side, you specify the rights to the content (Combined Delivery Lock or Separate Delivery Lock only).

- 5 Select the type of message you want to create:
  - **Forward Lock**
  - **Combined Delivery Lock**
  - **Separate Delivery Lock**
- 6 Under **Load Media Content**, click **Load Content..** to browse to the content file you want to wrap in the message. Select the content file and click **Open**.
- 7 For **Combined Delivery Lock** and **Separate Delivery Lock** only, enter a **Content-ID** for the content. This identifier must be unique to the media object, because it links the object to the rights.
- 8 For **Combined Delivery Lock** and **Forward Lock** only, for **Content-Transfer-Encoding**, select *one* of the following:
  - **Binary** – Leaves the content unencoded. Recommended for most content.
  - **Base64** – Encodes content by converting it from binary to 7-bit (ASCII) using a Base64 encoding scheme. (Recommended for use with mail programs that do not handle binary message content.)
- 9 For **Combined Delivery Lock** and **Separate Delivery Lock** only, under **Specify Rights**, enter the rights you want to associate with the content. See [Working with digital rights](#) on page 73.



10 Click **Show Message** to display the DRM message on the SDK.

For Separate Delivery Lock, only the content is sent to the SDK. You must click **Push Rights** to send the rights separately.

## Working with digital rights

You can set these permissions only when you create a Combined Delivery or a Separate Delivery message:

- **Play** – controls permissions for audio files (for example, .mp3).
- **Execute** – controls permissions for executables (for example, .jar or .exe).
- **Print** – controls permissions for printing, if the mobile phone is equipped to perform wireless printing (for example, .jpeg or .gif).
- **Display** – controls permissions for viewing images (for example, .jpeg or .gif).

The way you set each permission is identical. The rights you set are typically dictated by the content and the way you want the phone user to use the content.

If you do not enable a permission, the phone user will not have any rights to that permission.

To enable a permission:

- 1 On the left side of the DRM window, make sure you've selected one of the following:
  - **Combined Delivery Lock**
  - **Separate Delivery Lock**
- 2 On the right side of the DRM window, click the tab of the permission you want to enable:
  - **Play**
  - **Execute**
  - **Print**
  - **Display**
- 3 Make sure **Enable <permission> rights** is checked.
- 4 Check the following constraints you want to apply and enter a value for each one:
  - **Count** – enter the number of times the content can be used.
  - **Start Time** – enter the date and time after which the content can be used.
  - **End Time** – enter the date and time after which the content can no longer be used.
  - **Interval** – enter the maximum period of time during which the content can be used, beginning from the first use.

A summary of the permissions and constraints appears under **Rights Summary** as you enter the rights.

## Loading DRM rights on the SDK

When you work with DRM messages, you can push digital rights files directly from NMIT or by using **File>Open** in the SDK.

When the SDK receives a Forward Lock, Combined Delivery, or Separate Delivery content message, it displays a message asking you to select the folder to which you want the file downloaded. A list of folders where you can store content, or the combination of content and the rights to it, automatically displays.

When the SDK receives the rights part of a Separate Delivery message, it displays a message that the activation key has been received.

You can select **Options** to **Save**, **Discard**, or get **Details** about the rights. If you save the rights, the content to which the rights apply is activated.

## Working with an SMS message

With SMS, you can send text messages of up to 160 characters. Before you start the SDK instances you plan to use for SMS messages, make sure you've started NCF Lite 1.2. See [Simulating Wireless Connectivity between SDKs](#) on page 61.

## Creating an SMS message on the SDK

To create and send an SMS message directly on the SDK:

- 1 From the SDK, go to the idle screen by clicking the SDK **End Call** soft key one or more times. You know you are on the idle screen when you see **Go to**, **Menu**, and **Names** on the display.
- 2 Select **Menu>Messaging>Create message>Text message**.
- 3 Use the keyboard or click the keys on the first SDK keypad to enter the phone number of the second SDK instance to which you want to send the message.
- 4 Enter a text message.
- 5 Select **Send**.
- 6 Enter and send additional text messages or select **Exit** to exit.

## Receiving an SMS message on the SDK

When the second SDK receives the message, the SDK idle screen displays a note that a message has been received. You know you are on the idle screen when you see **Go to**, **Menu**, and **Names** on the display. If the idle screen does not display, the message goes into your inbox.

To process the message from the idle screen:

- 1 Select **Show** to display the message.

- 2 Select **Options** to **Delete**, **Forward**, or **Move** the messages to a folder in the SDK.

## Working with Bluetooth

With Bluetooth, you can send image files, from one SDK instance to another. Before you start the SDK instances you plan to use for Bluetooth file exchange, make sure you've started NCF Lite 1.2 and that the Bluetooth symbol is visible in the handset menus. See [Simulating Wireless Connectivity between SDKs](#) on page 61.

### Sending a file between SDK instances using Bluetooth

To send an image file between two SDKs:

- 1 From the SDK, go to the idle screen by clicking the **SDK End Call** soft key one or more times. You know you are on the idle screen when you see **Go to**, **Menu**, and **Names** on the display.
- 2 Select **Menu>Gallery>Images**, and highlight the image you want to send.
- 3 Select **Options>Send>Via Bluetooth...**If you have not named the device, the SDK suggests a name for it.
- 4 The SDK searches for available instances and displays the ones that have enabled Bluetooth.
- 5 Highlight the desired device instance number and click **Select...**The SDK sends the file to the chosen instance.  
Note that you have to either accept or reject the transfer from the target SDK instance.
- 6 The SDK displays a confirmation message if the transfer is successful. Click **OK**.

### Receiving a file from an SDK instance through Bluetooth

When the second SDK receives the image, the SDK idle screen displays a note prompting you to either accept or reject the transfer. Select **Accept** to start the transfer.

When the second SDK receives the file, the SDK idle screen displays a note that an image has been received. You know you are on the idle screen when you see **Go to**, **Menu**, and **Names** on the display. If the idle screen does not display, the image goes into your gallery.

To process the image from the idle screen:

- 1 Select **Show** to display the message.
- 2 Select **Options** to **Delete**, **Edit image**, or **Move** the image to a folder in the SDK.

## Importing a UI theme

A Series 40 UI theme package provides custom visual and audio motifs for mobile phones. The motifs can include a ring tone, color scheme, and images for a wallpaper, background, and screen saver.

To create a theme, create a theme package (.nth file) using the Nokia Series 40 Theme Studio product available at <http://www.forum.nokia.com>. For instructions on creating and saving a theme package, see *Nokia Series 40 Theme Studio User's Guide*.

To load a UI theme on the SDK:

- 1 Perform *one* of the following actions:
  - Select **File>Open**. Enter the path and name of the local .nth file you want and click **OK**. This copies the theme file to the SDK instance's E/Themes directory.
  - Select **File>Open URL**. Enter the URL of the .nth file you want on the website and click **OK**. This copies the theme file to the SDK instance's E/Themes directory.
  - Using the SDK Browser, navigate to a website that hosts the .nth file you want.
- 2 To apply the theme, select **Yes** in the SDK.  
The UI theme is applied to the mobile phone.

## What's next?

You can develop MIDlets, as described in [Using the SDK for MIDlet Development](#) on page 77.

Or, you might want to explore running the SDK from a command line. See [Running the SDK from a Command Line](#) on page 85.

---

# 7

## Using the SDK for MIDlet Development

A MIDlet is a Java application that conforms to the MIDP specification. MIDP applications are portable across mobile phones and can handle differences in screen size and button layout, as well as the low memory and small footprint of a mobile phone. Typically, a MIDlet is a game or application that you download from the Internet and use on a mobile phone. A MIDlet can be made up of:

- JAD file, a small file that describes the attributes of the larger JAR file. One function of the JAD file is to let a mobile phone determine whether it has the capability to run the JAR file prior to downloading it. A JAD file can contain information, such as:

```
MIDlet-Name: Bounce
MIDlet-Vendor: Nokia
MIDlet-Version: 1.0.1
MIDlet-Jar-Size: 25978
MIDlet-Jar-URL: http://domain/directory/Bounce.jar
MIDlet-1: MIDletName, MIDletIcon, MIDletMainClassName
```

- JAR file, which contains the actual preverified MIDlet. A JAR file contains the class, image, and sound files gathered into a single file and compressed for faster downloading to your SDK or handset.
- JAR manifest (`manifest.mf`), which contains information similar to the information in the JAD file. The JAR manifest is located within the JAR file and is available only after you've downloaded the JAR file.

## Setting up the SDK to run a MIDlet

To run a MIDlet on the SDK, you can set up persistent settings that:

- Control how MIDlets access the network. See [Setting up network access](#) on page 23.
- Control whether MIDlets are provisioned. See [Setting MIDP provisioning and execution options](#) on page 25.
- Set the KVM heapsize that controls how much memory you can have when the MIDlet runs. See [Setting the KVM heapsize](#) on page 27.
- Invoke more than one MIDlet on one SDK instance. See [Setting up to reuse an SDK instance to run MIDlets](#) on page 28.
- Provide event tracing and control the MIDlet speed. See [Setting MIDP tracing and speed options](#) on page 28.

If you are planning to send SMS messages between MIDlets that are running between SDKs, see [Simulating Wireless Connectivity between SDKs](#) on page 61.

## About OTA provisioning and direct loading of a MIDlet

Typically, you create a MIDlet with an IDE, such as Eclipse or Sun NetBeans, and run it on the SDK. For information about installing the SDK for use with IDEs, see *Series 40 5th Edition SDK Installation and Configuration Guide*.

You can run the MIDlet on the SDK with direct loading or over-the-air (OTA) provisioning. Direct loading is the default. Directly loading a MIDlet:

- Provides a faster, less restrictive environment in which to test and debug MIDlets during the early phases of development.
- Lets you develop a MIDlet from within an IDE during which you edit and compile code and immediately display the results on the SDK.
- Is a good way to test how a user runs a MIDlet that is already loaded on a handset.

OTA Provisioning runs the MIDlet file through the JAM. Provisioning a MIDlet can be more constraining but truer to what a user experiences when using a handset.

Provisioning a MIDlet:

- Lets you test how a user downloads a MIDlet from an Internet site.
- Provides MIDlet validation – a series of integrity checks that ensure the SDK can run the MIDlet. Based on the information in the JAD file, the JAM on the SDK determines whether the SDK can run the MIDlet.
- Saves the MIDlet in the computer file system, so it will be there the next time you start the SDK instance.

Here is a comparison of running a MIDlet on the SDK with OTA provisioning and direct loading:

MIDlet trait	Direct loading	OTA provisioning
MIDlet installed in the SDK instance's Applications folder	No	Yes
Maximum size of JAD file	10240 bytes (10KB)	10240 bytes (10KB)
Maximum size of JAR file	5MB (maximum value) 5MB (default value) Value can be changed in Tools>Preferences>MIDP	1MB
Maximum MIDlet data size	1MB	1MB

## Direct loading of a MIDlet

To directly load a MIDlet, set the preference to direct download (see [Setting MIDP provisioning and execution options](#) on page 25) and perform *one* of the following actions:

- Use **File>Open** or the SDK handset menus to load a local JAD or JAR file. See [About loading content on the SDK standalone application](#) on page 34.
- Use the command line without the `-xjam` option to open a local JAD or JAR file, for example: `emulator bounce.jar`. See [Running the SDK from a Command Line](#) on page 85.
- Send a MIDlet directly to the SDK from an IDE.

## OTA Provisioning a MIDlet

To load a MIDlet using OTA provisioning, set the MIDlet file provisioning preference to OTA provisioning (see [Setting MIDP provisioning and execution options](#) on page 25) and perform *one* of the following actions:

- Use **File>Open** or the SDK handset menus to load a local JAD or JAR file. See [About loading content on the SDK standalone application](#) on page 34.
- Use the command line to open a local JAD or JAR file, for example, `emulator bounce.jar`. See [Running the SDK from a Command Line](#) on page 85.

Alternatively, you can run a MIDlet using OTA provisioning and *one* of the following options, regardless of the preference setting:

- Use **File>Open URL** or the SDK handset menus to load a JAD or JAR file from a web server. However, you might have to set up a proxy server (see [Setting up network access](#) on page 23). Any MIDlet that is downloaded using HTTP is OTA provisioned, regardless of the SDK's provisioning settings.
- Use the command line interface and the `-Xjam` option to load a local JAD or JAR file, for example, `emulator -Xjam:install=mylocalmidlet.jad`.

## MIDP differences between the handset and the SDK

This section describes the MIDP differences between the handset and the SDK as they relate to:

- MIDlet connectivity
- Security
- JSR 135 audio/video

### MIDlet connectivity differences

The handset supports MIDlet connectivity using the serial cable and infrared port. If a MIDlet in the handset calls the `System.getProperty(microedition.commports)` method, the handset returns `COM0, IR0`. However, if the same MIDlet runs in the SDK, the SDK returns `COM0`, even though it does not simulate the serial or infrared communications paths.



## MIDP security differences

The handset can modify the MIDP security communication function group for a downloaded MIDlet using **Menu>Applications>Games>Application Access>Communication>Connectivity**. This option is not available on the SDK. You can modify the connectivity setting on the SDK using **Tools>Preferences>MIDP>Set Permissions**. However, this setting is only available for MIDlets downloaded using direct loading (see [About OTA provisioning and direct loading of a MIDlet](#) on page 78) and does not affect MIDlets that have already been downloaded and saved in the computer file system.

## JSR 135 audio/video differences

The handset supports numerous audio formats. See [What audio content types does the SDK support?](#) on page 41. The SDK supports most of these formats; however, the handset supports playing AAC audio and the SDK does not. Also, Real Time Streaming Protocol (RTSP) is not supported in the SDK.

The handset supports audio capture from its microphone and FM radio and stores it in the `audio/amr` format. Note that the SDK simulates recording from its microphone using pre-recorded content installed with the SDK. The SDK does not support recording from the FM Radio.

The handset API supports playing 3GPP, MPEG, and MPEG4 video. See [What video content types does the SDK support?](#) on page 42. The SDK simulates playing video, but does not show the video image in the SDK LCD display area. Instead, the SDK displays a reference video image, while the video is playing.

The handset API supports capturing still images and video clips from its camera. Note that the SDK simulates still image capture using pre-recorded content installed with the SDK. The SDK does not support video clip capture.

## Using the Series 40 5th Edition SDK with Eclipse 3.2.2

To use Eclipse with the SDK, you must install the SDK, Eclipse and EclipseME as well as configure Eclipse. For more information, see *Series 40 5th Edition SDK Installation and Configuration Guide*.

## Creating a new project for the MIDlet

To use the SDK with Eclipse, you create a project in Eclipse and import the MIDlet source code into it. Create a MIDlet package, which includes the JAD and JAR files that you can test on the SDK.

To create a new project that contains your MIDlet:

- 1 In Eclipse, select **File>New>Project** to access the New Project wizard.
- 2 In the New Project wizard, select **Java>J2ME Midlet Suite**.

- 3 Click **Next** and enter the project name.
- 4 Click **Next** and select the Nokia SDK you want to use.
- 5 Click **Next**. Make sure the Java settings for **Source**, **Project**, **Libraries**, **Order and Export**, and **Default Output Folder** are as they need to be for the project.
- 6 Click **Finish** to create the project. The name of the new project appears in the **Package Explorer** window.

## Creating or importing classes for the MIDlet

To create and build classes in Eclipse:

- 1 Select **New>Other>J2ME Midlet** to create a new class.
- 2 Enter the information for the class and click **Finish**.

To import the source and resources of the MIDlet into the project you've just created:

- 1 In the Eclipse's **Package Explorer** window, right-click on the project name and select **Import**. The **Import** window appears.
- 2 Navigate to the directory where the source file is located and select the source file.
- 3 In the **Into folder** field, enter the source file's name or browse to the source file. Select the folder where you want to import the file and click **Finish**.

## Creating the MIDlet application package

To create and run JAD and JAR files on the SDK:

- 1 In **Package Explorer** window, right-click the desired project and select **J2ME->Create Package**. Eclipse creates the JAD and JAR files under the project, in **deployed** folder.

## Testing the MIDlet

To test how the MIDlet works on the SDK:

- 1 In **Package Explorer** window, right-click the desired project and select **Run as->Run**. The Run window opens.
- 2 Right-click on your project , select Run as >Run option
- 3 In the left-side pane, double-click **Wireless Toolkit Emulators**. A new configuration opens in the right-side pane.
- 4 In the **Name** field, specify a name. The default is **New\_configuration**.
- 5 In **Executable** field, select your Midlet class.
- 6 Select **Emulation** tab, under heading **Device** select **Nokia SDK**.
- 7 Click **Apply**.

- 8 Click **Run**. You can view the MIDlet, as it executes. Messages that result from the compilation and running of the MIDlet display on the Eclipse console.

## Debugging the MIDlet

To debug the MIDlet:

- 1 Right-click on the project name in the **Eclipse Package Explorer** window and select **Debug As>Debug...**
- 2 In the **Debug** window that appears, enter the project name under **Project** and browse to the main class under **Main class**.
- 3 Click the **Nokia SDK Plug-in** tab and select the SDK to use.
- 4 Click **Debug**.

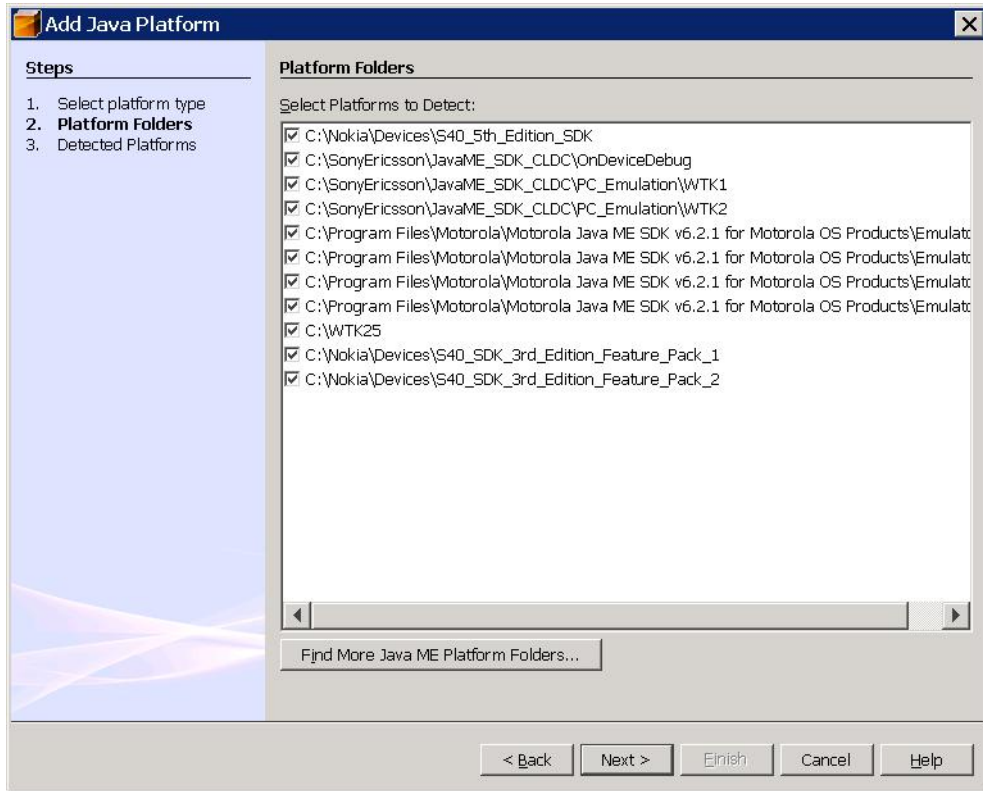
Eclipse runs the MIDlet on the SDK and displays a debug view.

## Using the Series 40 5th Edition SDK with Sun NetBeans

To run the SDK with Sun NetBeans, you need to configure a NetBeans project to use the SDK. To configure a project:

- 1 In NetBeans, create a new project or use an existing project.
- 2 From the left panel (project panel) of the NetBeans main window, right-click on the MIDlet project and select **Properties** to display the project properties.

- 3 Click **Platform** in the left panel. Select the SDK from the Emulator Platform drop-down menu, as shown in the following NetBeans window:



- 4 Click OK.

After configuring the NetBeans project, you can proceed with any operation on NetBeans. The operation uses the SDK you've chosen. For more information, see *Series 40 5th Edition SDK Installation and Configuration Guide*.

## Additional information about MIDlets

The SDK contains information about the MIDP Java APIs that it supports. To view this information, select **Help>User's guide>Java API Documentation** on the SDK menus.

Several sample Java MIDlets are installed with the SDK in the `examples` subdirectory at the path where you installed the SDK. Additional information about MIDlets is available at <http://www.forum.nokia.com>. To view this information, select **Help>MIDlet Samples** on the SDK menus.

## What's next?

You might want to explore running the SDK from a command line. See [Running the SDK from a Command Line](#) on page 85.

---

# 8

## Running the SDK from a Command Line

Although anyone can use the SDK through the Windows command line, you'll be more likely to use the CLI, if you are developing browser or message content. If you are developing MIDlets, you'll likely use command line options within a Java IDE, for example, to trace MIDlet events.

Configuration options you set on the command line override those you set in the **Tools>Preferences** dialog box but are not persistent between instances of the SDK. If you need settings to be persistent, use the settings in the **Tools>Preferences** dialog box.

## Accessing the command line prompt

To access the command line prompt:

- 1 Select **Start menu>Run**.
- 2 Enter `cmd` in the **Run** dialog box.
- 3 Navigate to the `\bin` folder of the folder where you installed the SDK.
- 4 Enter the command for starting the SDK, which is explained in the following section.

## About the command line syntax

The syntax for the command that starts the SDK is:

```
executable -option command ContentFile
```

The following table summarizes the components of the syntax:

Component	Description
executable	<p>The SDK supports two executables:</p> <ul style="list-style-type: none"><li>• <code>emulator.exe</code> — Supports options that are generally used for Java MIDlet development.</li><li>• <code>sdk.exe</code> — Supports options that are generally used for message and browser web page development.</li></ul> <p>See <a href="#">About the executables</a> on page 87.</p>
-option	<p>Options generally specify configuration settings for the SDK, such as the language it uses in each SDK instance and what events to trace. Most options work with <code>emulator.exe</code> and <code>sdk.exe</code>. However, some options work only with one of the executables. Those options are noted.</p> <p>See <a href="#">About options</a> on page 87.</p>
command	<p>The commands either load content or close an instance. You can have only one command per command line. The command and any options that go with it must be the last item in the command before the content file name. Separate the command from any other element with spaces.</p> <p>See <a href="#">About commands</a> on page 93.</p>
content file	<p>Load this file on the SDK.</p> <p>See <a href="#">Working with content types</a> on page 38.</p>

For example:

<code>emulator.exe</code>	<code>-Xontop -Xlanguage:Finnish</code>	<code>load mymidlet.jad</code>
executable	options (2)	command with content file

## About the executables

The SDK supports two executables (`emulator.exe` and `sdk.exe`). IDEs automatically decide which executable to use.

### About emulator.exe

Typically, an IDE that supports MIDlet development automatically uses `emulator.exe` when it uses the SDK as a device. The `emulator.exe` conforms to the Universal Emulator Interface (UEI) 1.0.2 specifications.

A developer of MIDlets is unlikely to use `emulator.exe` on the command line, except to start the monitoring of the behavior of a MIDlet. Options that you can use only with `emulator.exe` are noted in the option listing under [About options](#) on page 87. The SDK ignores options and commands that `emulator.exe` does not support.

### About 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. With this executable, you can work within a shell (see [Working with a CLI shell to repeatedly send commands](#) on page 95) and arrange to have one instance of the SDK send a message to another instance (see [Creating and sending an MMS message directly on the SDK](#) on page 67).

Options that you can use only with `sdk.exe` are noted in the option listing under [About options](#) on page 87.

Options and commands that `sdk.exe` does not support cause the entire command to fail, and an error message to be posted.

## About options

The SDK interprets all text preceded by a dash (-) as an option. If a value contains one or more space characters, the value must be enclosed between quotation marks (“”).

Options starting with -X are extensions to CLI standards. For example, the following command starts the SDK, specifying it should always remain on top of the other windows, and that it should display menu names in Finnish:

```
emulator -Xontop -Xlanguage:Finnish
```

Some options work only with `emulator.exe` or `sdk.exe`. These options are specifically noted with the name of the executable that supports them. Options that work with both executables are simply listed.

## Getting Information about the CLI options

To display a summary of the command line options in the Series 40 5th Edition SDK, enter `emulator -help` or `sdk -help`. The command displays a list of the options that work with the executable in the command window. The `-help` option must be the only option in a command.

### Options for managing SDK instances

You can use the following options to manage SDK instances:

Option	Description	Use with
<code>-Xnew</code>	Creates a new SDK instance. If you are using <code>-Xuse</code> in the same command, confirm that the instance ID is not in use. This will start the new instance with the desired instance ID. If the instance is already in use, a new SDK instance is not started.	<code>emulator.exe</code>
<code>-new</code>	Creates a new SDK instance. Do not use, if you are using <code>-use</code> in the same command.	<code>sdk.exe</code>
<code>-Xuse:&lt;instance ID&gt;</code>	Directs a command to the existing instance you specify instead of starting a new instance. For example: <code>emulator -Xuse:6267001</code> If you are using <code>-Xnew</code> in the same command, confirm that the instance ID is in use to direct the command to it. If the instance is not in use, this will start a new instance with the entered instance ID.	<code>emulator.exe</code>
<code>-use:&lt;instance ID&gt;</code>	Directs a command to the existing instance you specify instead of starting a new instance. For example: <code>sdk -use:6267001</code> Do not use, if you are using <code>-new</code> in the same command.	<code>sdk.exe</code>



Option	Description	Use with
<code>-shell</code>	Creates a shell in which you can repeatedly load commands to the SDK instance within the shell. See <a href="#">Working with a CLI shell to repeatedly send commands</a> on page 95.	<code>sdk.exe</code>
<code>-qinst</code>	Displays the instance IDs of all running instances. Use <code>-qinst</code> as the sole option in a command line.	<code>sdk.exe</code>

## Options for general use

With the exception of `-version`, the following options take effect only when the result of the command creates a new instance:

Option	Description	Use with
<code>-Xhttp_proxy:</code> <code>&lt;proxy address and port number&gt;</code>	Specifies the address and port number of the HTTP proxy the HTTP protocol will use when making network requests. Specify the value in the following format: <code>address:port_number</code> For example: <code>113.14.176.11:8080</code> or <code>proxy.domain:8080</code> If the port number is not specified, the port 8080 is used.	
<code>-Xlanguage:&lt;language&gt;</code>	Specifies the language that the SDK should use for its menu names.	
<code>-version</code>	Displays version information about the SDK.  This option does not create a new instance.	

Option	Description	Use with
-Xinbox: <directory name>	<p>Specifies the directory that the new instance of the SDK checks for MMS files. For example:</p> <pre>sdk -Xinbox:d:\MMSFiles</pre> <p>When you place MMS files in the directory, the SDK displays and then deletes them.</p> <p>For <code>emulator.exe</code>, the directory name can be absolute or relative to the program. For <code>sdk.exe</code>, the directory name must be absolute. If the directory does not exist, the SDK creates it.</p> <p>This function is available only through the CLI.</p>	
-Xontop	<p>Specifies that the SDK window should remain on top of all other open windows and must be minimized or moved to get to the windows below it.</p> <p>Default value: Off</p>	
-Xoutbox: <directory name>	<p>Specifies the directory to which the SDK sends messages as MMS files. For example:</p> <pre>sdk -Xoutbox:d:\MMSSentFiles</pre> <p>For <code>emulator.exe</code>, the directory name can be absolute or relative to the program. For <code>sdk.exe</code>, the directory name must be absolute. If the directory does not exist, the SDK creates it.</p> <p>This function is available only through the CLI.</p>	

## Options for tracing Java runtime

Trace output displays within the output window of an IDE or within the SDK DOS Console when you run the SDK as a standalone application. For information on how to set up the tracing, see [Setting up tracing to file](#) on page 28.

The following tracing options apply only to Java output:

Option	Prints	Use with
-Xverbose:allocation	Memory allocation trace	<code>emulator.exe</code>

Option	Prints	Use with
-Xverbose:class	Class loading trace	emulator.exe
-Xverbose:gc	Garbage collection trace	emulator.exe
-Xverbose:gcverbose	Verbose garbage collection trace	emulator.exe
-Xverbose:threading	All threads in the system	emulator.exe
-Xverbose:classverbose	Verbose class loading trace	emulator.exe
-Xverbose:verifier	Verifier trace	emulator.exe
-Xverbose:stackmaps	Stackmap trace	emulator.exe
-Xverbose:bytecodes	Byte code trace	emulator.exe
-Xverbose:method	Method trace call	emulator.exe
-Xverbose:methodcallsverbose	Verbose method call trace	emulator.exe
-Xverbose:stackchunks	Stackchunk trace	emulator.exe
-Xverbose:frames	Frame trace	emulator.exe
-Xverbose:exceptions	Exception trace	emulator.exe
-Xverbose:events	Event trace	emulator.exe
-Xverbose:monitors	Monitor trace	emulator.exe
-Xverbose:networking	Networking trace	emulator.exe
-Xverbose:	Enable all Java traces	emulator.exe
-Xverbose:all		

## Options for general management of Java output

You can use the following options to perform general work with MIDlets:

Option	Description	Use with
-classpath <directories and JAR files>	Specifies the directories and JAR files to be searched for classes; the classpath may include both library and application class directories and JAR packages.	emulator.exe
-Xdescriptor:<JAD file>	Runs an application using the specified JAD file.	emulator.exe
-Xdebug	Runs the MIDlet in debugging mode.	emulator.exe

Option	Description	Use with
-Xdevice:<device name>	Run an application on the SDK specified by the device name.	emulator.exe
-Xheapsize:<heapsize>	Specifies the heapsize (in bytes) available for MIDlets. The value may be specified in kilobytes by appending k to the number, or in megabytes by appending M to the number. The valid value range is 0 to 10 megabytes. If the specified value is greater than the valid range (more than 10 megabytes), the heapsize is set to the maximum value.	emulator.exe
-Xjam:install=<JAD_filename>	Makes the SDK load the specified MIDlet from the hard drive using the MIDlet OTA provisioning method instead of the direct loading method.	emulator.exe
-Xquery	Prints information about the SDK. This option is not valid when used with the -Xjam option.	emulator.exe
-Xrunjdpw:address=<port>	Specifies the TCP/IP port number between the emulator KVM and the debug proxy.	emulator.exe
-Xrunjdpw:transport=dt_socket	Specifies the transport mechanism used to communicate with the debugger. Only dt_socket is supported.	emulator.exe
-Xrunjdpw:server=Y	Starts the debug agent as a server. The debugger connects to the specified port.	emulator.exe
-Xrunjdpw:suspend=Y	Suspends the VM immediately after establishing a connection to the debugger. Only the value Y is supported.	emulator.exe

Option	Description	Use with
-Xsecurity_domain: <domain name>	Specifies the security domain for the MIDlet. Domain name can have one of the following values: <ul style="list-style-type: none"><li>• Trusted</li><li>• Untrusted</li><li>• Maximum</li><li>• Minimum</li></ul>	emulator.exe

## About commands

You can use the following two commands:

- `load` - works with `emulator.exe` and `sdk.exe`
- `close` - works only with `sdk.exe`.

## About the Load command

The `load` command specifies the filename or URL to be loaded on an SDK instance.

The syntax of the load command is:

```
load <value>
```

The values you can use are:

- MMS filename
- WML filename
- URL string
- JAD filename
- JAR filename

For example:

- To load a local WML file:  

```
sdk load C:\welcome.wml
```
- To load a local MIDlet:  

```
emulator load mymidlet.jar
```
- To load a remote WML file:  

```
sdk load http://server.domain/welcome.wml
```

You can imply the `load` command when you are not working within a shell (see [Working with a CLI shell to repeatedly send commands](#) on page 95). To imply the `load` command, omit `load` and type the name of the content file. For example:

```
sdk mymidlet.jar
```

## About the Close command

You can use the `close` command only with `sdk.exe`.

The `close` command works in two ways, depending on whether you are within a shell or outside a shell (see [Working with a CLI shell to repeatedly send commands](#) on page 95):

- When you use `close` by itself within a shell, `close` terminates the shell. Running instances are left running. To terminate the instance, you must use `close` with `-exit`. See [-exit option](#) on page 94.
- The `close` command by itself outside a shell does nothing, unless you use it with either `-exit` or `-exitall`. See [-exit option](#) on page 94 and [-exitall option](#) on page 94.

The syntax of the `close` command is:

```
close -option
```

### **-exit and -exitall options**

Unlike other options, `-exit` and `-exitall` work only with the `close` command and, when used, must immediately follow the command. For example:

```
sdk close -exit  
or  
sdk close -exitall
```

#### *-exit option*

Use the `-exit` option with the `close` command to close one of the following:

- Within a shell: the SDK instance within the shell.
- Outside a shell: the SDK instance you specify with `-Xuse` or the SDK instance with the highest number.

#### *-exitall option*

Use the `-exitall` option with the `close` command outside a shell to close all running SDK instances. For example, the following command closes all open SDK instances:

```
sdk close -exitall
```

## Working with a CLI shell to repeatedly send commands

Use the `-shell` option with `sdk.exe` to create a session where you can repeatedly send commands to the one instance without having to recreate the instance with every command. You can attach a shell to an instance that was created through a standalone SDK or through an application, such as Eclipse. In this example, the instance 6267001 is already running and is used to create a shell for:

```
sdk -shell -use:6267000      ← Create a shell using instance
                             6267000.

%%-->> instance : 6267000

-> load http://server.domain ← Load this website into instance
    welcome.wml               6267000 in the shell.

%%-->> load : ok

-> load c:\welcome.wml      ← Load this WML file into
                             instance 6267000 in the
                             shell.

%%-->> load : ok

-> close                    ← Close the shell. Instance
                             6267000 continues to
                             exist.

%%-->> detaching
```

To close both the shell and the instance within the shell, use the `close` command with the `-exit` option:

```
-> close -exit
%%-->> exiting
```

In the example above, the `close` command closes the shell session, and the `-exit` option closes the instance within the session.





---

# Index

## Symbols

.nth file **65, 76**

## Numerics

1099 RMI registry port **24**

2D Vector Graphics API **10**

## A

Adobe Go Live CS2 **16**

API Java information **84**

## B

Bluetooth

file exchange **75**

receiving **75**

sending **75**

wireless connectivity, and **61**

Bluetooth APIs **10**

Browser History, diagnostics **56**

Browser Source, diagnostics **52**

bytecode instructions, speed **30**

## C

canvas

speed **30**

Canvas window **36**

-classpath option **91**

CLDC **10**

CLI **13**

close command **94**

close, command **93**

color rendering **12**

Combined Delivery Lock message (DRM) **70**

command **86**

close **93, 94**

load **93**

command line

executables, list of **87**

general use options **89**

how to repeatedly send

commands **95**

interface **13**

managing SDK instances

options **88**

online information about **88**

options **87**

parameters **93**

prompt **86**

prompt, accessing **86**

syntax **86**

tracing Java output **90**

tracing options **13**

commands, from keyboard **37**

Connected Limited Device

Configuration **10**

connectivity **61**

constraints, DRM **73**

content

loading **34**

loading with File>Open **34**

loading with File>Open URL  
**34**

loading with handset menus  
**35**

on the SDK **13**

previewing with Adobe Golive  
**35**

responses **50**

types **38**

viewing in Canvas **36**

content types

browser, and **40**

file types **38**

in-line **41**

video **42**

WMLScript **40**

## D

Default directory **58**

default handset behavior **25**

Diagnostics window **13, 46**

Browser History view **56**

Browser Source view **52**

enabling, disabling **21**

Log view **57**

MIDP view **51**

shaping **47**

summary of views **48**

Traffic view **48**

turning off **45**

Digital Rights Management **10, 70**

digital rights message **65**

digital rights, defined **73**

direct loading **26**

DirectLoaded directory **59**

DRM **65**

combined delivery lock **70**

constraints **73**

creating messages **71**

creating with NMIT **71**

defined **70**

digital rights, working with **73**

forward lock **70**

loading rights **74**

message types **70**

permissions **73**

rights summary **73**

separate delivery lock **71**

## E

Eclipse **81**

SDK, and **81**

emulator.exe **87**

events **42**

executable **86**

emulator.exe **87**

sdk.exe **87**

execution options **27**

exiting SDK **17**

## F

feedback to Nokia **17**

File connection API **10**

file system **58**

file types of **38**

Forward Lock (DRM) **70**

## G

General tab, preferences **20**

generating events **42**

## H

Help menu **16**

HTTP

- cache **24**
- requests, responses **48**

**HTTPS**

- requests, responses **48**

## I

**IDE 16**

- tracing options **29**

in-line content types **41**

instance directory **58**

instance ID, changing **20**

instance number prefix **20**

instance reuse **28**

internal RMI Registry port **24**

## J

**JAD 77**

- file size **79**

**JAR**

- file size **26, 79**

Java API information **84**

Java Application Descriptor **77**

Java MIDlet samples **84**

Java Technology for Wireless Industry **10**

JSR 120 **62**

JSR 205 **61**

## K

keyboard shortcuts **37**

kilobyte virtual machine **51**

**KVM 51**

- heapsize **27**
- heapsize status **52**
- speed **30**

## L

load, command **93**

loading content **34**

Log view, diagnostics **57**

## M

Macromedia

- Flash Lite **10**

main window **12**

management **91**

message

- creating **65**
- MMS **66**
- Push **50**

SMS **74**

message content development **65**

**MIDlet**

- canvas speed **30**
- defined **77**
- development **77**
- direct loading **26**
- directly loading **79**
- execution options **27**
- KVM **51**
- KVM speed **30**
- MIDP, and **77**
- network access **23**
- preferences **23**
- provisioning **25, 79**
- running on SDK **78**
- samples **84**
- setting up SDK **78**
- speed controls **30**
- tracing option **14**
- tracing options, IDE **29**

**MIDP 9**

- diagnostics **51**
- Monitor tab, preferences **29**

**MMS message 10, 48, 65**

- content types in **66**
- creating on SDK **67**
- creating with NMIT **69**
- defined **66**
- processing on SDK **69**

Mobile Information Device Profile **9**

mobile internet content **35**

Mobile Media API **10**

Multimedia Messaging Service **10**

## N

**NCF 10, 61**

- icon **62**
- shutting down **62**
- starting **62**

**NetBeans 83**

- network access **23**
- network security **26**
- network security level **26**
- Networking tab, preferences **23**
- new option **88**

**NMIT 16, 35**

- DRM, and **71**
- MMS message **69**

Nokia Connectivity Framework **10, 61**

Nokia Mobile Internet Toolkit **16, 35**

Nokia, feedback to **17**

non-provisioning

compared to provisioning **79**

## O

### OMA

Client Provisioning **10**

DRM **10**

### options

- about **87**
- classpath **91**
- configuration settings **86**
- for managing SDK instances **88**
- for tracing Java output **90**
- new **88**
- option **86**
- qinst **89**
- shell **89**
- version **89**
- Xdebug **91**
- Xdescriptor **91**
- Xdevice **92**
- Xheapsize **92**
- Xhttp\_proxy **89**
- Xinbox **90**
- Xjam **92**
- Xlanguage
  - 89
- Xnew **88**
- Xontop **90**
- Xoutbox **90**
- Xquery **92**
- Xrunjdpw **92**
- Xsecurity\_domain **93**
- Xuse **88**
- Xverbose **90**

## P

PC file system **58**

permissions, DRM **73**

permissions, security domain **26**

Persistent Record Management Store emulation **27**

phone number prefix **20**

phone number, changing **20**

phone sound **22**

PIM API **10**

predefgallery directory **59**

predefjava directory **59**

preferences **15**

- General tab **20**

- HTTP cache **24**

- introduction **19**

- JAR file size **26**

- KVM heapsize **27**

MIDlet **23**

MIDlet canvas speed **30**

MIDlet execution options **27**

MIDlet KVM speed **30**

MIDlet speed controls **30**

MIDP Monitor tab **29**

network access **23**

network security **26**

Persistent Record Management Store emulation **27**

phone number prefix **20**

phone sound **22**

provisioning mode **25**

proxy **23**

tracing options **29**

user security prompts **27**

### previewing

- mobile internet content **35**

- WML files **35**

### provisioning

- benefits **79**

- compared to non-provisioning **79**

- MIDlets **79**

- mode **25**

proxy server **23**

Push message **50**

## Q

-qinst option **89**

quick tour of SDK **11**

## R

reuse SDK instance **28**

RMI Registry port **24**

## S

### SDK

- as standalone **34**

- color, and **12**

- content **13**

- DRM rights, loading **74**

- executable **86**

- information about **16**

- instance reuse **28**

- main window **12**

- MIDlet running **78**

- MMS processing **69**

- overview of **9**

- quick tour **11**

- requests **50**

- SMS, creating **74**

- SMS, receiving **74**

- starting **11**
- startup behaviors **20**
- Sun NetBeans **83**
- using **33**
- window size **21**
- with IDE **15, 45**
- sdk.exe **87**
- Security and Trust Services API **10**
- Security Domain **26**
- Separate Delivery (DRM) **71**
- Separate Delivery Lock message (DRM) **71**
- shell option **89**
- Short Message Service **74**
- SMS
  - wireless connectivity, and **61**
- SMS message **49, 74**
  - between SDKs **62**
  - creating on SDK **74**
  - defined **74**
  - fragment **49**
  - NCF, and **62**
  - received **51**
  - receiving on SDK **74**
  - sent **50**
- startup behaviors **20**
- Sun NetBeans **83**

## T

- TCP stream data **48, 51**
- theme package **76**
- tracing
  - tracing to file **28**
- tracing options **14, 29**
  - within IDE **29**
- Traffic view
  - diagnostics **48**
  - displaying events **50**
  - events recorded **48**

## U

- UAProf **10**
- UDP datagram **48, 51**

- UI theme **65**
- UI theme package, defined **76**
- User Agent Profile **10**
- user security prompts **27**

## V

- version option **89**
- video content types **42**

## W

- WAP Push message **48**
- Web Services API **10**
- wireless connectivity **61**
  - Bluetooth **61**
  - SMS **61**
- Wireless Messaging API **10**
- WML and Adobe Golive **35**
- WML files **35**
- WML variables, Diagnostics **54**
- WMLScript **40**

## X

- Xdebug option **91**
- Xdescriptor option **91**
- Xdevice option **92**
- Xheapsize option **92**
- XHTML Mobile Phone Profile **10**
- Xhttp\_proxy option **89**
- Xinbox option **90**
- Xjam option **92**
- Xlanguage
  - option **89**
- Xnew option **88**
- Xontop option **90**
- Xoutbox option **90**
- Xquery option **92**
- Xrunjdwp options **92**
- Xsecurity\_domain option **93**
- Xuse option **88**
- Xverbose options **90**

---

**NOKIA**

---

**NOKIA**