Java J2ME™ for
Sony Ericsson mobile phones

April 2005
Preface

Purpose of this document


Readers who will benefit from this document include:

- Software developers
- Corporate buyers
- IT professionals
- Support engineers
- Business decision-makers

It is assumed that the reader is familiar with Java.
Sony Ericsson Developer World

On www.SonyEricsson.com/developer, developers will find documentation and tools such as phone White Papers, Developers Guidelines for different technologies, SDKs and relevant APIs. The website also contains discussion forums monitored by the Sony Ericsson Developer Support team, an extensive Knowledge Base, Tips & Tricks, example code and news.

Sony Ericsson also offers technical support services to professional developers. For more information about these professional services, visit the Sony Ericsson Developer World website.

Document conventions

Products

Sony Ericsson mobile phones are referred to in this document using generic names as follows:

<table>
<thead>
<tr>
<th>Generic names</th>
<th>Sony Ericsson mobile phones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Series</td>
<td></td>
</tr>
<tr>
<td>Z1010</td>
<td>Z1010</td>
</tr>
<tr>
<td>K700</td>
<td>K700i, K700c</td>
</tr>
<tr>
<td>S700</td>
<td>S700i, S700c, S710a</td>
</tr>
<tr>
<td>K500</td>
<td>K500i, K506c, K508i, K508c</td>
</tr>
<tr>
<td>F500</td>
<td>F500i</td>
</tr>
<tr>
<td>Z500</td>
<td>Z500a</td>
</tr>
<tr>
<td>V800</td>
<td>V800, Vodafone 802SE</td>
</tr>
<tr>
<td>K300</td>
<td>K300i, K300c, K300a</td>
</tr>
<tr>
<td>J300</td>
<td>J300i, J300c, J300a</td>
</tr>
<tr>
<td>K750</td>
<td>K750i, K750c, D750i</td>
</tr>
<tr>
<td>W800</td>
<td>W800i, W800c</td>
</tr>
<tr>
<td>Z800</td>
<td>Z800i</td>
</tr>
<tr>
<td>K600</td>
<td>K600i</td>
</tr>
</tbody>
</table>
Terminology and abbreviations

API
Application Programming Interface.

CLDC
Connected Limited Device Configuration. A J2ME configuration for mobile phones.

DRM
Digital Rights Management.

GSM
Global System for Mobile Communications. GSM is the world’s most widely used digital mobile phone system, now operating in over 100 countries around the world, particularly in Europe and Asia-Pacific.

HTTP
HyperText Transfer Protocol.

IDE
Integrated Development Environment.

J2ME
Java 2 Platform, Micro Edition. A Java platform targeting "micro" devices with small processors and memory capacities, such as mobile phones, communicators and PDAs.

J2SE

JSR
Java Specification Request.

Mascot Capsule®
Mascot Capsule Micro 3D Engine is software that renders 3D objects in real-time on a display screen of an embedded device, portable game unit or cellular handset.

MIDP
Mobile Information Device Profile. A J2ME profile connected to the CLDC for mobile phones.

MMAPI
Mobile Media Application Programming Interface.

OMA
Open Mobile Alliance.

SDK
Software Development Kit. A collection of tools used to develop applications.

SMS
Short Message Service. Allows messages of up to 160 characters to be sent and received in a mobile phone via the network operator’s message centre.

URI
Uniform Resource Identifier. URIs are short strings that identify online resources: documents, images, downloadable files, services, and electronic mailboxes, for example. URIs use a variety of naming schemes and access methods, such as http, ftp, mailto and telnet, to make resources available.

URL
Uniform Resource Locator. See URI.

WAP
Wireless Application Protocol.

WMA
Wireless Messaging API.

WTK
Wireless Toolkit.
Trademarks and acknowledgements

Java and all Java-based marks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.
End-user license agreement for Sun Java™ J2ME™.

1 Restrictions: Software is confidential copyrighted information of Sun and title to all copies is retained by Sun and/or its licensors. Customer shall not modify, decompile, disassemble, decrypt, extract, or otherwise reverse engineer Software. Software may not be leased, assigned, or sublicensed, in whole or in part.

2 Export Regulations: Software including technical data, is subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. Customer agrees to comply strictly with all such regulations and acknowledges that it has the responsibility to obtain licenses to export, re-export, or import Software. Software may not be downloaded, or otherwise exported or re-exported (i) into, or to a national or resident of, Cuba, Iraq, Iran, North Korea, Libya, Sudan, Syria (as such listing may be revised from time to time) or any country to which the U.S. has embargoed goods; or (ii) to anyone on the U.S. Treasury Department’s list of Specially Designated Nations or the U.S. Commerce Department’s Table of Denial Orders.

3 Restricted Rights: Use, duplication or disclosure by the United States government is subject to the restrictions as set forth in the Rights in Technical Data and Computer Software Clauses in DFARS 252.227-7013(c) (1) and FAR 52.227-19(c) (2) as applicable.

Borland, the Borland Logo and JBuilder are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries.

NetBeans is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

The Bluetooth word mark and logos are owned by the Bluetooth SIG, Inc. and any use of such marks by Sony Ericsson is under license.

Nokia is a registered trademark of Nokia Corporation.

Mascot Capsule is a registered trademark of HI Corporation.

ARM and Jazelle are registered trademarks of ARM Limited.

RealAudio and RealVideo are trademarks or registered trademarks of RealNetworks, Inc.

Other product and company names mentioned herein may be the trademarks of their respective owners.
Java Verified™ Program for the J2ME platform

The Java Verified™ Program uses the results of the Unified Testing Initiative launched by the leading mobile phone manufacturers and Sun Microsystems.

The Java Verified program gives developers a direct way to application testing and to the market. Testing Providers all over the world, covering different regions, languages and price structures, are authorized by the Java Verified program to undertake testing on behalf of the program. A developer selects one of these providers to complete the testing of their application.

An application that successfully meets both the program guidelines and passes the testing process is permitted to use the Java Powered logo. The logo is provided by Sun Microsystems, at its own discretion, on a non-exclusive license basis. An application that passes the Java Verified program testing is digitally signed so that potential distributors can be assured of its integrity and authenticity.

Once the developer has successfully passed the testing process, their application has the opportunity to be promoted and showcased in the on-line catalogs of all Java Verified Member Companies, as well as the commercial catalogs of participating operators.

More information on the Java Verified Program is available at the Java Verified Web site (www.javaverified.com).
## Document history

<table>
<thead>
<tr>
<th>Change history</th>
<th>Version</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-12-05</td>
<td>R1A</td>
<td>First edition</td>
</tr>
<tr>
<td>2003-12-11</td>
<td>R1B</td>
<td>Minor updates in technical specifications.</td>
</tr>
<tr>
<td>2004-03-30</td>
<td>R2A</td>
<td>Document updated to comply with the latest software version of the Z1010 phone. Information about the K700 series and Z500 series added. 3D API information added.</td>
</tr>
<tr>
<td>2004-07-19</td>
<td>R4A</td>
<td>Information about the K500 series added.</td>
</tr>
<tr>
<td>2004-11-23</td>
<td>R5A</td>
<td>Information about the V800 series added.</td>
</tr>
</tbody>
</table>
Contents

Java J2ME™ ..............................................................................................................10
Java Platforms ........................................................................................................11
MIDP 2.0 support ................................................................................................12
WMA (JSR 120) ....................................................................................................12
MMAP (JSR 135) ..................................................................................................13
  Audio support ....................................................................................................14
  Video support ....................................................................................................14
3D APIs ........................................................................................................................15
PDA optional packages (JSR 75) .........................................................................15
  PIM optional package ......................................................................................15
  File Connection optional package ..................................................................15
Bluetooth API (JSR 82) .....................................................................................16
Memory ...................................................................................................................16
The navigation key ..................................................................................................17
Simultaneous key presses ......................................................................................17
Command types ......................................................................................................17
Error messages .......................................................................................................18
Sony Ericsson J2ME SDK support .........................................................................18
  Permission settings ..........................................................................................19
  Security Configuration ......................................................................................20
Download and installation ....................................................................................22

Appendix A
Phone specifications ..............................................................................................23
  Screen and memory specifications ...................................................................24
  Java specifications ..........................................................................................26
  Camera specifications .....................................................................................28
  Font sizes ..........................................................................................................29
  Key mapping ......................................................................................................30

Appendix B
Java programming issues .....................................................................................32
  Hints for developing MIDlets ..........................................................................32
  Writing efficient applications .......................................................................32
  Low-level MIDP user interface ....................................................................33
  Memory usage ...................................................................................................33
  Java heap ..........................................................................................................33
  Video RAM areas ............................................................................................34
  Retrieving the IMEI number ..........................................................................35
  Network APIs ..................................................................................................35
  Secure sockets and HTTPS connections .....................................................36
  JAD/manifest attributes ..................................................................................37
  Vodafone JAD attributes ................................................................................37
  JSR 75 implementation ....................................................................................38
  PIM API ............................................................................................................38
  File Connection API .......................................................................................41
  Tips for using the JSR 82 ................................................................................42
  Local device ......................................................................................................42
  Device discovery ...............................................................................................43
Games .................................................................................................................43
Querying system properties ....................................................................................44
Supported classes ...............................................................................................44
System.getProperty(String Key) calls ..................................................................44
Bluetooth Local device properties (JSR 82) .........................................................46
Implementation specific properties in JSR 184 ...................................................46

Links and references ..............................................................................................48

Appendix C
Sony Ericsson J2ME SDK ......................................................................................50
Features ..................................................................................................................51
Sony Ericsson J2ME SDK integration in JBuilder 2005 .......................................51
Sony Ericsson J2ME SDK integration in NetBeans 4 ..........................................52
The mobile phones covered in this document support the MIDP 2.0 and CLDC 1.1 specifications. They also have Java support for sending and receiving SMS (via WMA 1.1), playing audio and video, as well as taking snapshots from the built-in cameras (via MMAPI 1.1). The basic MIDP 2.0 features, such as life cycle, memory handling etc, are the same as for the MIDP 1.0 environment. More information about MIDP 1.0 in Sony Ericsson mobile phones is available at Sony Ericsson Developer World (www.SonyEricsson.com/developer/java). MIDP 1.0 applications developed for the T61x, T628/T630, and Z60x phones should also execute on Sony Ericsson MIDP 2.0 supported phones.
Java Platforms

To help content developers to adapt applications for a multitude of phones, Sony Ericsson has adopted a Java platform strategy for its mobile phones. Sony Ericsson has two main J2ME platform paths for its phones, one for its Symbian OS based smartphones and another one for its feature and mass-market phones, that is, the non-Symbian OS based phones. Normally each platform version is used in several phone models.

A list of Sony Ericsson J2ME platform versions for the phones in the scope of this document can be found below. Some platform features are optional, that is, configurable. For example, the Bluetooth API’s (JSR 82) are only enabled for a specific phone when the phone actually supports Bluetooth wireless technology.

JP = Java Platform

<table>
<thead>
<tr>
<th>Features</th>
<th>Phones</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP-2</td>
<td>CLDC 1.1, MIDP 2.0, JTWI (JSR 185), JSR 120, JSR 135, Nokia UI API 1.1</td>
</tr>
<tr>
<td>JP-3</td>
<td>CLDC 1.1, MIDP 2.0, JTWI (JSR 185), JSR 120, JSR 135, Nokia UI API 1.1, JSR 184, Mascot Capsule Ver. 3</td>
</tr>
<tr>
<td>JP-4</td>
<td>CLDC 1.1, MIDP 2.0, JTWI (JSR 185), JSR 120, JSR 135, Nokia UI API 1.1, JSR 184, Mascot Capsule Ver. 3</td>
</tr>
<tr>
<td></td>
<td>Optional: VSCL 2.0</td>
</tr>
<tr>
<td>JP-5</td>
<td>CLDC 1.1, MIDP 2.0, JTWI (JSR 185), JSR 120, JSR 135, Nokia UI API 1.1, JSR 184, Mascot Capsule Ver. 3, JSR 75</td>
</tr>
<tr>
<td></td>
<td>Optional: JSR 82</td>
</tr>
</tbody>
</table>
MIDP 2.0 support

The mobile phones covered in this document are MIDP 2.0 and JTWI 1.0 compliant. For a list of protocols, formats, memory size, display size etc. supported by the MIDP 2.0 implementation in the phones, see “Appendix A Phone specifications” on page 23, which contains technical specifications for each device.

The MIDP 2.0 specification contains a number of optional features of which the following are supported:

- PushRegistry Alarm and PushRegistry SMS. In JP-4 and JP-5, PushRegistry CBS is also supported.
- Signed MIDlets as specified in JTWI 1.0.
- TCP and UDP server sockets as specified in MIDP 2.0.
- PlatformRequest supports the tel, http and https schemes.
  When the method is invoked with the tel scheme, the native phone application is accessed and the user can initiate a voice or video call, or send a message to the given phone number.
  A PlatformRequest invocation for http/https initiates downloading of the given URI, for example a Java application, image etc. For http/https URIs referencing to WAP pages, the web browser is invoked. The Java application is then left in the background until the phone call/download/web session is completed, after which it is resumed.
- GameCanvas.getKeyStatus() supports the detection of several simultaneous keys. See also “Simultaneous key presses” on page 17
- TextBox and TextField with input constraints ANY, EMAILADDR and URL support the character set specified in JTWI 1.0.
- PNG images with colour depth of 1, 2, 4, 8, 16, 24 and 32 bits per pixel are supported.
- The maximum number of application-created threads is limited only by the amount of available memory.
- A TextField or TextField object with input constraint TextField.PHONENUMBER allows the user to select a phone number from the phone’s phone book, as specified in JTWI.

WMA (JSR 120)

The Wireless Messaging API v 1.1 (JSR 120) is supported. GSM SMS is supported in all mobile phones covered in this document, while GSM Cell Broadcast (CBS) is only supported in JP-4 and JP-5. MIDP 2.0 security has been added to the Open connection, Send and Receive functions, as specified in WMA 1.1, http://www.jcp.org/en/jsr/detail?id=120.

The Sony Ericsson J2ME SDK provides support for developing WMA MIDlets. This includes API documentation, support for compiling WMA MIDlets and debugging these MIDlets using any of the mobile phones covered in this document.

Per Appendix A, GSM SMS Adapter, implementations of the GSM SMS adapter must support at least three concatenated short message segments. The mobile phones covered in this document exceed the minimum requirement, allowing MIDlets to send and receive SMS messages of up to ten segments in length.

The 3GPP specification for SMS specifies the port numbers 16000-16999 as available for applications. It is recommended that Java developers use non-reserved port numbers within this range. WMA has a system list of restricted port numbers which may not be used by the Java application. In addition to the port
numbers restricted in the WMA specification, the mobile phones covered in this document also reserve the ports listed in the table below. If a Java application attempts to use any of the restricted and/or reserved ports, an exception will be thrown.

<table>
<thead>
<tr>
<th>Port number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Internal system use</td>
</tr>
<tr>
<td>650</td>
<td>General obex</td>
</tr>
<tr>
<td>2948-2949</td>
<td>WAP</td>
</tr>
<tr>
<td>5505</td>
<td>PM ringtone (Nokia Smart Messaging)</td>
</tr>
<tr>
<td>5506</td>
<td>PM Logo (Nokia Smart Messaging)</td>
</tr>
<tr>
<td>5507</td>
<td>PM Icon (Nokia Smart Messaging)</td>
</tr>
<tr>
<td>5514</td>
<td>Picture message (Nokia Smart Messaging)</td>
</tr>
<tr>
<td>9200-9207</td>
<td>WAP</td>
</tr>
<tr>
<td>16733</td>
<td>Calendar</td>
</tr>
<tr>
<td>16987</td>
<td>Email notification</td>
</tr>
<tr>
<td>16988</td>
<td>Email account setting</td>
</tr>
<tr>
<td>49996-49997</td>
<td>WAP provisioning</td>
</tr>
<tr>
<td>49999</td>
<td>WAP provisioning</td>
</tr>
</tbody>
</table>

**MMAPI (JSR 135)**

The MMAPI support in the mobile phones covered in this document provides access to audio and video playback, as well as the capture of images with the phone’s camera. For a list of supported data formats for each phone, please refer to “Java specifications” on page 26. The JSR 135 specification can be downloaded from [http://www.jcp.org/en/jsr/detail?id=135](http://www.jcp.org/en/jsr/detail?id=135).

Players can be created from:

- Java streams
- DataSources
- URIs with “http://”, “https://” or “capture://video”
Audio support

See also “Java specifications” on page 26.

In **JP-2** and **JP-3**, a maximum of 16 audio players can exist at the same time in Started state at the Java level. The number of players that can produce audio in parallel is limited by the device hardware. Simple tones can be generated in parallel to any of the supported audio formats, but no other parallel audio playback is supported.

In **JP-4** and **JP-5**, the number of simultaneously started players is limited only by available memory. These phones also support more advanced mixing. One player can play a waveform audio file (.WAV or ADPCM) with a sample rate of 8 or 16 kHz in parallel to another player playing a MIDI file.

The MidiControl, supported in **JP-5**, allows control of a maximum of 16 MIDI channels when playing MIDI.

The following controls are implemented:

- VolumeControl
- ToneControl
- StopControl
- MidiControl (**JP-5** only)

The following controls are **not** implemented:

- TempoControl
- PitchControl
- RateControl

Video support

See also “Java specifications” on page 26

**Note**: Video playback is not supported in **JP-2**.

Only one video player can exist at a particular time. The video player can display its contents in a **Canvas** or in an **Item** on a **Form**.

The snapshot functionality is only supported for taking a picture with the device’s built-in camera. Access to the camera snapshot functionality follows the security policy specified in MIDP 2.0, i.e. it is categorized as a “Multimedia recording”. (See “The Recommended Security Policy for GSM/UMTS Compliant Devices” in the MIDP 2.0 specification.)

When a phone call is received while running a Java application that uses the native camera, the Java reference to the native camera is released. Once the phone call is terminated, the Java application will regain focus and an END_OF_MEDIA_EVENT is sent to the Java application. It is then up to the Java application whether to restart the camera or not.

The following control is implemented:

- VideoControl
3D APIs

*Note*: *JP-2* does not support the 3D APIs.

*JP-3*, *JP-4* and *JP-5* support real-time 3D graphics rendering. These platforms support two different 3D graphics APIs. Mascot Capsule Micro3D version 3 and version 4 (JSR 184 compliant) sets are supported. For more information on the implementation of the 3D APIs, please refer to the Developers Guideline *Java 3D*, available at [www.sonyericsson.com/developer/java](http://www.sonyericsson.com/developer/java).

PDA optional packages (JSR 75)

*Note*: Only *JP-5* support the JSR 75 API.

The PDA optional packages for the J2ME platform (JSR 75) consist of two separate APIs, one for accessing PIM data and one for file system access.

PIM optional package

The PIM (Personal Information Management) API is standardized in the JSR 75 specification, which can be downloaded from [http://www.jcp.org/en/jsr/detail?id=75](http://www.jcp.org/en/jsr/detail?id=75). The following describes shortly the implementation in the Sony Ericsson mobile phones where the API is supported.

In Sony Ericsson mobile phones the PIM API handles:

- Contacts (ContactList)
- Calendar (EventList)
- Tasks (ToDoList)

For more details on the Sony Ericsson implementation of the PIM package, please refer to “Appendix B Java programming issues” on page 32.

File Connection optional package

The File Connection API is standardized in the JSR 75 specification, which can be downloaded from [http://www.jcp.org/en/jsr/detail?id=75](http://www.jcp.org/en/jsr/detail?id=75). The following describes shortly the implementation in the Sony Ericsson mobile phones where the API is supported.

In general, Java applications can access the same folders, subfolders and files as the built-in File manager application, both in the phone's internal memory and on an inserted memory card. The following folders and all contained subfolders and files are accessible via the API:

- `<file://c/>` (internal memory file root)
Note: The folders **Games** and **Themes** are **not** available via the File Connection API.

For more details on the Sony Ericsson implementation of the package, please refer to Appendix B, “JSR 75 implementation” on page 38.

---

**Bluetooth API (JSR 82)**

See also “Java specifications” on page 26.

**Note:** Only **JP-5** support the Bluetooth API.

**JP-5** supports JSR 82, the standard Java API for Bluetooth. It provides the means for developers to create Bluetooth games and other applications as well as implement new Bluetooth profiles.

For example, the Bluetooth API offers developers the ability to:

- Create multiplayer games
- Connect to PCs from Java applications

For more information about the Bluetooth API, see “Java specifications” on page 26. The complete JSR 82 specification can be downloaded from the Java Community Pages, [http://www.jcp.org/en/jsr/detail?id=82](http://www.jcp.org/en/jsr/detail?id=82)

---

**Memory**

The mobile phones covered in this document utilize a number of different memory areas for user interface features, and for images in particular. The total amount of memory available varies depending on how much of this memory other native phone applications have currently allocated. If needed, and if memory is available, the Java heap grows dynamically up to about 1.5 MB. In general, a Java application with around 500 kB of image data should be executable.

For more information about memory in different phones, please see “Java specifications” on page 26. More information about memory allocation can be found in Appendix B, “Memory usage” on page 33.
The navigation key

The mobile phones covered in this document detect navigation key actions in the following manner:

- Two adjacent directions are simultaneously detected. If the navigation key is pressed in one of the four main directions, up, down, left or right, one event is delivered to the application. If the navigation key is pressed in a diagonal direction, two events are delivered to the application, for example one for “up” and one for “right”.
- Navigational changes are detected directly without having to go back to neutral position.

Simultaneous key presses

Support for simultaneous key presses enhances gaming experience. For example, a user can move around on the screen and shoot at the same time.

Most Sony Ericsson phones support simultaneous key presses. However, a MIDP developer can not take for granted that a certain device support simultaneous key presses in all possible combinations. Games and other applications should always be tested with the actual targeted hardware. Hardware emulators does not necessarily emulate simultaneous key presses properly.

In general, when two keys are pressed at the same time, the proper events are delivered to the application. When three or more keys are pressed in some combinations, extra key presses are detected. In other combinations, the third key is not detected at all. The general approach when more than two keys need to be detected at the same time, is to map the game keys (Fire, game A, game B, and so on) to actions that might occur at the same time as two or more other key presses.

More specific information for the simultaneous key press functions can be found in Appendix A, “Key mapping” on page 30.

Command types

The MIDP commands defined by an application are displayed on either the left selection key, the right selection key or they are placed in the “More” menu which is associated with the right selection key. Command type BACK is always mapped to the phone’s back button. Command type OK is generally mapped to the left selection key. It is recommended to always include BACK and OK commands. The command types are prioritized in the following order (from higher to lower):

- OK
- ITEM
- SCREEN
- BACK
- CANCEL
If a command of the type BACK exists it will be mapped to the device’s back key. It is also recommended to use the BACK command instead of the EXIT command for exiting the application, allowing the user to press the phone’s back key to exit. This conforms to the normal behaviour of Sony Ericsson mobile phone applications.

Of all the remaining commands (excluding the one mapped to back) the one with the highest priority is mapped to the left selection key. All other commands are mapped to the right selection key. If more than one command is to be mapped to the right selection key, a “More” option is displayed and a list of the commands appears when the user selects the right selection key.

### Error messages

Java exceptions that are not handled by the active MIDlet are dealt with by the Java environment. The following error messages are displayed to the user:

<table>
<thead>
<tr>
<th>Java exception</th>
<th>Error message displayed to user</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.io.IOException</td>
<td>Network failure</td>
</tr>
<tr>
<td>javax.microedition.io.ConnectionNotFoundException</td>
<td>Network failure</td>
</tr>
<tr>
<td>java.lang.ClassNotFoundException</td>
<td>Invalid application</td>
</tr>
<tr>
<td>java.lang.OutOfMemoryError</td>
<td>The application consumes too much memory</td>
</tr>
<tr>
<td>java.io.EOFException, java.io.UnsupportedEncodingException, java.io.UTFDataFormatException</td>
<td>Network data error</td>
</tr>
<tr>
<td>javax.microedition.rms.RecordStoreFullException</td>
<td>Application memory full</td>
</tr>
<tr>
<td>All other Exceptions or Errors</td>
<td>Application error</td>
</tr>
</tbody>
</table>

### Sony Ericsson J2ME SDK support

Development of Java applications for the mobile phones covered in this document is supported by the Sony Ericsson J2ME SDK. This includes PC emulation based on a customized version of the Wireless Toolkit from Sun. For example, the screen size, colour depth and key inputs of the device are emulated.

The latest version of the Sony Ericsson J2ME SDK is available for download at [www.sonyericsson.com/developer/java](http://www.sonyericsson.com/developer/java).
Security policy for Sony Ericsson phones

All of the devices described in this document comply with the JSR185 Java Technology for the Wireless Industry (JTWI) specification and MIDP 2.0 recommended security policy. For a detailed description of the installation and security rules, see Chapter 7, Security Policy for GSM/UMTS Compliant Devices of the JSR 185 specification (http://www.jcp.org/en/jsr/detail?id=185). A number of APIs are categorized as “restricted”. Usage can result in costs for the user (traffic charges), inappropriate use may potentially affect the user data integrity or cause disturbance to other parties. The following tables describe the specific security configuration implementation on the Java platforms JP-2, JP-3, JP-4 and JP-5.

Permission settings

The following table lists definitions of permission settings:

<table>
<thead>
<tr>
<th>Permission setting</th>
<th>Screen description</th>
<th>Definition</th>
<th>Standard description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes, Ask Once per Session</td>
<td>Ask the first time an application requests this function, then retain this setting for the remainder of the session.</td>
<td>Session (Yes)</td>
</tr>
<tr>
<td>2</td>
<td>Yes, Always Ask</td>
<td>Ask every time the application requests access to this function</td>
<td>One-shot (Yes)</td>
</tr>
<tr>
<td>3</td>
<td>No, Ask Later</td>
<td>Do not accept the request to the function at this time, however next time this function is needed during this session please ask again.</td>
<td>One-shot (No)</td>
</tr>
<tr>
<td></td>
<td>(Runtime only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>No Access</td>
<td>Do not grant permission to this function during this session and do not ask again.</td>
<td>Session (No)</td>
</tr>
<tr>
<td></td>
<td>No, Never Grant</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Runtime Only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Not Allowed</td>
<td>This function is not granted access by the operator. The Application is not allowed access to this function and the user is unable to modify this setting.</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>Allowed</td>
<td>This function is always granted access by the operator</td>
<td>Blanket</td>
</tr>
<tr>
<td>7</td>
<td>Yes, Never Ask Again</td>
<td>User Defined - never ask again for permission, permission always granted.</td>
<td>Blanket (Yes)</td>
</tr>
<tr>
<td></td>
<td>Blanket Permission</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Security Configuration

This table lists permission settings per functionality and security domain:

<table>
<thead>
<tr>
<th>Functionality/Domain</th>
<th>Untrusted</th>
<th>Trusted 3rd party</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Network</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>javax.microedition.io.HttpConnection</td>
<td>1, 2, 4</td>
<td>1, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.HttpsConnection</td>
<td>1, 2, 4</td>
<td>1, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.datagram</td>
<td>1, 2, 4</td>
<td>1, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.datagramreceiver datagram server (w/o host)</td>
<td>1, 2, 4</td>
<td>1, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.socket</td>
<td>1, 2, 4</td>
<td>1, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.serversocket server socket (w/o host)</td>
<td>1, 2, 4</td>
<td>1, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.ssl ssl</td>
<td>1, 2, 4</td>
<td>1, 7, 4</td>
</tr>
<tr>
<td><strong>All App Auto-Start</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>javax.microedition.io.PushRegistry</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td><strong>Messaging- Wireless Messaging API - JSR 120</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>javax.wireless.messaging.sms.send</td>
<td>2, 4</td>
<td>2, 4</td>
</tr>
<tr>
<td>javax.wireless.messaging.sms.receive</td>
<td>2, 4</td>
<td>2, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.sms</td>
<td>2, 4</td>
<td>2, 4</td>
</tr>
<tr>
<td>javax.wireless.messaging.cbs.receive <strong>(JP-4 and JP-5 only)</strong></td>
<td>2, 4</td>
<td>2, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.cbs <strong>(JP-4 and JP-5 only)</strong></td>
<td>2, 4</td>
<td>2, 4</td>
</tr>
<tr>
<td><strong>PIM and File Connection APIs (JSR 75, <strong>JP-5 only</strong>)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>javax.microedition.pim.ContactList.read</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.pim.ContactList.write</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.pim.EventList.read</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.pim.EventList.write</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.pim.ToDoList.read</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.pim.ToDoList.write</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.file.read</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
<tr>
<td>javax.microedition.io.Connector.file.write</td>
<td>1, 2, 4</td>
<td>2, 7, 4</td>
</tr>
</tbody>
</table>
Note: Unsigned MIDlets are not allowed to:

- open datagram connections on ports 9200, 9201 or 9203.
- open socket connections on ports 80, 443 or 8080.
- open SSL connections on port 443.

The security domain is determined at installation as follows:

- If the midlet suite is unsigned, then it will be installed in the "Untrusted" domain.
- If the midlet suite was signed using a certificate granted by a trusted third party such as Verisign or Thawte, then it will be installed in the "Trusted 3rd party" domain. Operators maintain control of their certification process.
- A signed midlet suite is not installed if certificate verification fails, for example when a midlet suite, signed by one operator, is attempted to install on a phone issued by another operator. In other words, operator signatures are not generic, but are specific to devices provided by each individual operator.

The digital certificate embedded in the JAD and the signed JAR file are verified for authenticity and date validity at install time according to chapter 4 of the JSR 118 specification (http://www.jcp.org/en/jsr/detail?id=118). This assures data integrity and vendor identity.

Certificates in Sony Ericsson mobile phones

The table below lists "factory installed" certificates in Sony Ericsson phone models/series. The table is valid for the first released version of the different phones, later releases may in some cases contain more certificates.

<table>
<thead>
<tr>
<th>Phone model/series</th>
<th>Certificates</th>
<th>UTI from GeoTrust (Java Verified)</th>
<th>Verisign</th>
<th>Thawte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z500</td>
<td></td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>S700i, S700c</td>
<td></td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>S710a</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V800</td>
<td></td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>K300</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Download and installation

The typical distribution mechanism for MIDP applications is over the air (OTA) via WAP or HTTP. The JAD and or JAR file(s) are accessible via the Internet and users may access either file. After downloading via the phones Browser application, installation is automatic.

**Note:** When JAR files are downloaded OTA via a WAP gateway, the file size may be limited by the network operator.

In the case of a signed midlet, the user must access the JAD file, because the signature is in it. The JAD file is read and the URL property in the file is used to access the JAR file. Transferring the JAR file via Bluetooth, IR or serial cable will not work, since this method will only work for unsigned midlets.

Signed midlets may also be installed using the correct JAD file with the Sony Ericsson J2ME SDK via the DeviceExplorer, ejava.exe command line tool or by right-clicking the file and select the “Install on Device” item.

A signed midlet can be installed on a phone with no UTI root certificate, by removing the following JAD/Manifest attributes and the corresponding values before installation:

- MIDlet-Certificate-1-1:
- MIDlet-Jar-RSA-SHA1:

A list of JAD attributes supported in MIDP2 compliant Sony Ericsson mobile phones can be found in Appendix B, see “JAD/manifest attributes” on page 37.

Java applications can be installed on the memory card as well as in internal memory in **JP-4** and **JP-5** mobile phones with memory cards. **Note:** This is not possible in the S700 series.

**To install a MIDlet on the memory card:**

- Transfer the application files (JAR/JAD) to the directory \mssemc\media files\other in the mobile phone’s files system.
- From the phone’s main menu select File manager (Data folder) and browse to the application in the other directory. Select **Install**.
Appendix A
Phone specifications

In this appendix the technical specifications are listed for the mobile phones covered in this Developers Guideline.

Note: market/customer variations in the specifications may exist.
Screen and memory specifications

Screen sizes are specified as Width x Height (pixels).

<table>
<thead>
<tr>
<th>Specification/Phone</th>
<th>Z1010</th>
<th>K700</th>
<th>S700</th>
<th>K500, F500, Z500</th>
<th>V800, Z800</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Screen</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Screen size</td>
<td>176x220</td>
<td>176x220</td>
<td>240x320</td>
<td>128x160</td>
<td>176x220</td>
</tr>
<tr>
<td>Fullscreen canvas</td>
<td>176x220</td>
<td>176x220</td>
<td>240x320</td>
<td>128x160</td>
<td>176x220</td>
</tr>
<tr>
<td>size</td>
<td>176x182</td>
<td>176x176</td>
<td>240x266</td>
<td>128x128</td>
<td>176x182</td>
</tr>
<tr>
<td>Pixel ratio (H:W)</td>
<td>1:1</td>
<td>1:1</td>
<td>1:1</td>
<td>1:1</td>
<td>1:1</td>
</tr>
<tr>
<td>Colour depth</td>
<td>65,536 (16 bit)</td>
<td>65,536 (16 bit)</td>
<td>262,144 (18 bit) a</td>
<td>65,536 (16 bit)</td>
<td>262,144 (18 bit) a</td>
</tr>
<tr>
<td>Transparency</td>
<td>Full (8 bit) alpha blending</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. RMS size</td>
<td>Limited only by the amount of available free storage.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory, storage</td>
<td>8 MB</td>
<td>40 MB</td>
<td>32 MB</td>
<td>10 MB Z500: 6 MB</td>
<td>V800 8 MB b</td>
</tr>
<tr>
<td></td>
<td>Z800 5 MB b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: The amount of memory available for Java applications depends on the free amount of internal memory in the mobile phone. Other contents, such as pictures, video clips and themes, use the same memory pool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java heap memory</td>
<td>512 kB - 1.5 MB (dynamic, depending on available memory)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Native video RAM availa-</td>
<td>Approx. max 500 kB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ble to Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) In Java only 65,536 colours (16 bit) can be used.

b) Java applications can be installed on the memory card as well as in internal memory.

To install a MIDlet on the memory card:

- Copy the application files (JAD/JAR) to the directory \mssemc\media files\other in the mobile phone’s files system.
- From the phone’s main menu select Data folder and browse to the application in the other directory. Select Install.
### Screen and memory specifications - continued

<table>
<thead>
<tr>
<th>Specification/Phone</th>
<th>K300, J300</th>
<th>K750, W800, K600</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Screen</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Screen size</td>
<td>128x128</td>
<td>176x220</td>
</tr>
<tr>
<td>Fullscreen canvas size</td>
<td>128x128</td>
<td>176x220</td>
</tr>
<tr>
<td>Non fullscreen canvas size</td>
<td>128x110</td>
<td>176x176</td>
</tr>
<tr>
<td>Pixel ratio (H:W)</td>
<td>1:1</td>
<td>1:1</td>
</tr>
<tr>
<td>Colour depth</td>
<td>65,536 (16 bit)</td>
<td>262,144 (18 bit)</td>
</tr>
<tr>
<td>Transparency</td>
<td>Full (8 bit) alpha blending</td>
<td></td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. RMS size</td>
<td>Limited only by the amount of available free storage.</td>
<td></td>
</tr>
<tr>
<td>Memory, storage</td>
<td>8 MB</td>
<td>K750 34 MBb K600 37 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The amount of memory available for Java applications depends on the free amount of internal memory in the mobile phone. Other contents, such as pictures, video clips and themes, use the same memory pool</td>
</tr>
<tr>
<td>Java heap memory</td>
<td>512 kB - 1.5 MB (dynamic, depending on available memory)</td>
<td></td>
</tr>
<tr>
<td>Native video RAM available to Java</td>
<td>Approx. max 500 kB</td>
<td></td>
</tr>
</tbody>
</table>

a) In Java only 65,536 colours (16 bit) can be used.

b) Java applications can be installed on the memory card as well as in internal memory.
   To install a MIDlet on the memory card:

- Copy the application files (JAD/JAR) to the directory `\mssemc\media files\other` in the mobile phone's files system.
- From the phone’s main menu select Data folder and browse to the application in the other directory. Select Install.
# Java specifications

The following table lists the Java characteristics of the phones covered in this document.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLDC version</td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>MIDP version</td>
<td>2.0</td>
<td>See also “MIDP 2.0 support” on page 12</td>
</tr>
</tbody>
</table>

**Supported image formats:**
- GIF87a, GIF89a, PNG v 1.0 (colour depth 1, 2, 4, 8, 16 bits per pixel), BMP v 3.x, WBMP level 0.

**Networking:**
- secure sockets, http 1.1, https. TLS 1.0 is also supported.

<table>
<thead>
<tr>
<th>JTWI (JSR 185) compliant</th>
<th>Yes, Release 1</th>
</tr>
</thead>
</table>

| MMAPI (JSR 135) | 1.1 | See also “MMAPI (JSR 135)” on page 13 |

**Supported Audio Content types:**
- audio/midi - MIDI (GM, GML and SP-MIDI)
- audio/x-wav - WAV (PCM)
- audio/x-tone-seq - JSR 135 tone sequence
- audio/mpeg - MP3 (MPEG-1 layer 3, MPEG-2 layer 3, MPEG 2.5 layer 3)
- audio/imelody - iMelody
- audio/amr - AMR
- audio/mp4a-latm - 3GP (MPEG-4 AAC LC)
- audio/x-pn-realaudio (.ra) - RealAudio®, ver. 8 (**K600** only)

**Supported Video Content types:**
- video/mp4v-es - 3GP (MPEG-4 Visual Simple Profile Level 0)
- video/h263-2000 - 3GP (H.263 Baseline Profile 0 Level 10)
- video/x-pn-realvideo (.rm) - RealVideo®, ver. 8 (**K600** only)

**Supported Image (Camera) Content types:**
- image/jpeg - JPEG

| WMA (JSR 120) | 1.1 - GSM SMS | See also “WMA (JSR 120)” on page 12 |

**Note:** Video playback not supported in **JP-2**
### PDA Optional Packages for the J2ME Platform (JSR 75)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support</th>
<th>Comments</th>
</tr>
</thead>
</table>
| **Version 1.0** | | **Note:**
| **PIM API, supported package:** | | Only supported in **JP-5**
| - Javax.microedition.pim | | See also “PDA optional packages (JSR 75)” on page 15 and “JSR 75 implementation” on page 38|
| **PIM API, supported classes/interfaces:** | | |
| - Contact | | |
| - Event | | |
| - ToDo | | |
| - Serialization methods on PIM items | | |
| - Serialization of PIM items according to vCard 2.1/vCalendar 1.0 | | |
| **File connection API, supported package:** | | |
| - javax.microedition.io.file | | |

### Java Bluetooth API (JSR 82)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support</th>
<th>Comments</th>
</tr>
</thead>
</table>
| **Version 1.0 a** | | **Note:**
| **Supported packages:** | | Only supported in **JP-5**
| - javax.bluetooth | | See also “Bluetooth API (JSR 82)” on page 16|
| - javax.obex | | |
| **Supported connections:** | | |
| - L2Cap (btl2cap://) | | |
| - Serial Port Profile (btspp://) | | |
| - Generic Object Exchange Profile (btgoep://) | | |
| - irdaobex (irdaobex://) | | |
| **Not supported:** Push Registry | | |

### Java IR APIs

- No

### Java Serial APIs

- No

### OTA Recommended Practice

- Yes, MIDP 2.0 compliant

### Debug Interface

- KDWP

### Numeric Keys

- Yes (0-9, *, #)

### 8-way Directional Key with Select

- Yes (navigation key)

### Signed MIDlets

- Yes

### TCP Sockets

- Yes

### UDP Sockets

- Yes

### Java 3D

- Mascot Capsule Micro3D version 3
- Mascot Capsule Micro3D version 4 (JSR 184)

See also “3D APIs” on page 15

**Note:** Not supported in **JP-2**
Camera specifications

In the S700, K750, W800, and K600 the native camera application is designed for taking pictures with the phone in horizontal position. When a snapshot is taken in a Java application, the image is automatically rotated to match the image seen in the Java viewfinder.

The table below lists supported resolutions for snapshot images in Java applications. Java applications are restricted to use the listed resolutions, even if the camera itself supports other image formats.

<table>
<thead>
<tr>
<th>Phone/series</th>
<th>Resolution (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>60x100</td>
</tr>
<tr>
<td>Z1010, K700, K500, Z500, F500, K300</td>
<td>•</td>
</tr>
<tr>
<td>S700</td>
<td>•</td>
</tr>
<tr>
<td>V800, Z800</td>
<td>•</td>
</tr>
<tr>
<td>K750, W800</td>
<td>•</td>
</tr>
<tr>
<td>K600</td>
<td>•</td>
</tr>
</tbody>
</table>

*a* High resolution snapshots may not be possible to view in the Java application that took the picture, because of limited memory. However, it is possible to take the snapshot in the application and then process the created image object, for example save it as a file in the file system of the phone.
Font sizes

A font is specified by requesting a style, size and face. In Sony Ericsson mobile phones, the style and size attributes are supported, while the face attribute is ignored.

Note: Font attributes are only available for Java in the low-level UI (Canvas or GameCanvas objects).

Due to space restrictions, all styles are not supported for Chinese characters. Also, the SIZE_LARGE attribute gives the same size as SIZE_MEDIUM.

Font heights in pixels (including line space) are listed below:

<table>
<thead>
<tr>
<th>MIDP values</th>
<th>Z1010</th>
<th>K700</th>
<th>S700</th>
<th>K500</th>
<th>K300</th>
<th>K750</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F500</td>
<td>J300</td>
<td>K600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z500</td>
<td></td>
<td>W800</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V800</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Western characters</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE_LARGE</td>
<td>22 px</td>
<td>22 px</td>
<td>26 px</td>
<td>20 px</td>
<td>15 px</td>
<td>22 px</td>
</tr>
<tr>
<td>SIZE_MEDIUM</td>
<td>18 px</td>
<td>18 px</td>
<td>22 px</td>
<td>15 px</td>
<td>13 px</td>
<td>18 px</td>
</tr>
<tr>
<td>SIZE_SMALL</td>
<td>15 px</td>
<td>15 px</td>
<td>18 px</td>
<td>13 px</td>
<td>9 px</td>
<td>15 px</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chinese characters</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE_LARGE</td>
<td>18 px</td>
<td>22 px</td>
<td>26 px</td>
<td>15 px</td>
<td>15 px</td>
<td>22 px</td>
</tr>
<tr>
<td>SIZE_MEDIUM</td>
<td>18 px</td>
<td>22 px</td>
<td>26 px</td>
<td>15 px</td>
<td>15 px</td>
<td>22 px</td>
</tr>
<tr>
<td>SIZE_SMALL</td>
<td>15 px</td>
<td>18 px</td>
<td>22 px</td>
<td>13 px</td>
<td>13 px</td>
<td>18 px</td>
</tr>
</tbody>
</table>
Key mapping

Sony Ericsson mobile phones support the `keyPressed()`, `keyReleased()`, and `keyRepeated()` event delivery methods in class `Canvas`.

<table>
<thead>
<tr>
<th>Key</th>
<th>Constant value</th>
<th>MIDP key code</th>
<th>Game action</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-way select up</td>
<td>-1</td>
<td>UP</td>
<td></td>
</tr>
<tr>
<td>4-way select down</td>
<td>-2</td>
<td>DOWN</td>
<td></td>
</tr>
<tr>
<td>4-way select left</td>
<td>-3</td>
<td>LEFT</td>
<td></td>
</tr>
<tr>
<td>4-way select right</td>
<td>-4</td>
<td>RIGHT</td>
<td></td>
</tr>
<tr>
<td>4-way select press</td>
<td>-5</td>
<td>FIRE</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>42</td>
<td>KEY_STAR</td>
<td>GAME_C</td>
</tr>
<tr>
<td>#</td>
<td>35</td>
<td>KEY_POUND</td>
<td>GAME_D</td>
</tr>
<tr>
<td>0</td>
<td>48</td>
<td>KEY_NUM0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>49</td>
<td>KEY_NUM1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>KEY_NUM2</td>
<td>UP</td>
</tr>
<tr>
<td>3</td>
<td>51</td>
<td>KEY_NUM3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>KEY_NUM4</td>
<td>LEFT</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>KEY_NUM5</td>
<td>FIRE</td>
</tr>
<tr>
<td>6</td>
<td>54</td>
<td>KEY_NUM6</td>
<td>RIGHT</td>
</tr>
<tr>
<td>7</td>
<td>55</td>
<td>KEY_NUM7</td>
<td>GAME_A</td>
</tr>
<tr>
<td>8</td>
<td>56</td>
<td>KEY_NUM8</td>
<td>DOWN</td>
</tr>
<tr>
<td>9</td>
<td>57</td>
<td>KEY_NUM9</td>
<td>GAME_B</td>
</tr>
<tr>
<td>Left Selection key (Soft key).</td>
<td>-6</td>
<td>Only available in Fullscreen Canvas mode.</td>
<td></td>
</tr>
<tr>
<td>Right Selection key (Soft key).</td>
<td>-7</td>
<td>Only available in Fullscreen Canvas mode.</td>
<td></td>
</tr>
<tr>
<td>C key (Clear)</td>
<td>-8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Back key</td>
<td>-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operator key ^a</td>
<td>-10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

^a This key is not present in all phone models and is referred to with different names, depending on what function it is used for in the UI of the phone. The name and function of the key may also be customized for different operators.
Simultaneous keypress support (Z1010, S700, Z500)
Two keys pressed at the same time are properly detected. More than two simultaneous key presses can in some combinations generate extra key press events. In some three key sequences, the third key is not detected.

Simultaneous keypress support (K700, F500, K500, K300, and J300)
Two keys pressed at the same time are properly detected. More than two simultaneous key presses can in some combinations generate extra key press events. Pressing the navigation key in different directions and pressing other keys at the same time works with key 5 (Fire), key 7 (Game A) and key 9 (Game B). Using other keys together with the navigation key can generate false detections.

Simultaneous keypress support (JP-4 and JP-5)
Two keys pressed at the same time are properly detected. More than two keys pressed simultaneously works in most cases but some combinations of keys may generate extra key press events. Pressing the navigation key and other keys at the same time works with key 0, key 1, key 3, key 7 (GAME A), key 9 (GAME B), key *(GAME C) and key # (GAME D).
Appendix B
Java programming issues

This appendix contains some programming issues of interest for developers of Java MIDlets/applications for Sony Ericsson mobile phones.

Hints for developing MIDlets

Information specific for developing Java MIDlets for wireless devices may be found in Applications for Mobile Information Devices, a Sun white paper with helpful hints for application developers and user interface designers using the MIDP (http://java.sun.com/j2me/docs/pdf/midpwp.pdf). The book MIDP 2.0 Style Guide (Wagner, Bloch - Addison Wesley, 2003) contains practical guidelines for utilizing the features of MIDP 2.0. In addition, please refer to the FAQ section of the Sony Ericsson J2ME SDK release notes for more information about application development.

Writing efficient applications

Java MIDlets run on devices with limited screen sizes, memory and processing power. Reducing the number of created and destroyed objects will reduce memory usage and at the same time improve performance by reducing the time spent by the JVM for initialization and garbage collection of these objects.

Some recommendations for writing efficient applications:
Developers Guideline Java J2ME™ for Sony Ericsson mobile phones

- Make good use of static variables and avoid operations on String objects.
- Use the StringBuffer class for efficient manipulation of strings.
- Limit the use of inner classes and use an obfuscator to reduce class file size.
- Set object references to null as soon as they are no longer needed.
- Avoid unnecessary re-initialization of variables that are automatically set to 0 or null by the VM.
- Use synchronization sparingly. It is costly and is only needed in multi-threaded applications.
- Avoid loading the same image into memory more than once, since memory is consumed for each duplicate.
- Close network streams when finished with, in order to preserve resources.

Low-level MIDP user interface

An application using the low-level MIDP user interface will always have a Canvas object. The Canvas is implemented with double buffering to eliminate display flicker. The buffer is flushed when the paint() method returns.

Some recommendations for writing low-level UI applications:

- Only repaint the part of the Canvas to be changed, but always remember to paint what is requested of you to paint.
- The method startApp() is not only called when your midlet starts, but also when resuming after calling pauseApp().

Memory usage

Java MIDlets/applications allocates memory in several different memory areas. Memory problems most often occur with allocation of memory for images. In this section some issues concerning memory usage are covered.

Java heap

Java applications use two kinds of heap memory, plain Java heap and LAM (Large Array Memory). The LAM is shared with other processes on the phone. Standard Java objects and vectors of Java objects are always located on the Java heap. Arrays of primitive types (byte[], int[], float[]) however may be
put in the LAM if the plain Java heap is low on memory. Small arrays have a larger chance of ending up in the plain java heap, while large arrays more often are stored in the LAM. Images are also sometimes placed in LAM.

The size and configuration of the Plain heap size and the LAM size varies between phone models.

The size of LAM is not included in the values reported by `Runtime.freeMemory` and `Runtime.totalMemory`.

Some simple rules to make the most of phone memory:

1. Always release memory before reallocating it:
   ```java
   char[] v = new char[100];
   ... do stuff ...
   v = null; // by setting v to null the allocation below can re-use the memory.
   v = new char[200];
   ```

   The same schema goes for pictures, resources etc. for the phone to be able to re-use a image vector the image must first be released:
   ```java
   Object o = allocateMyResource(size);
   ... do stuff ...
   o = null; // Remove the reference to the resource so that it can be resued in the allocation below
   o = allocateMyResource(someOtherSize);
   ```

2. Allocate your objects first, then your primitive arrays and images.

Video RAM areas

To assure that Java MIDlets will not run out of memory due to use of graphics, the Sony Ericsson mobile phones covered in this document implement several memory areas for graphics. Graphics memory areas are used in the following order. If one area is full or an image to large to fit in the free space of one area, the next one is used instead.

1. One area of fast video RAM dedicated for graphics storage.
2. Another video RAM area, with somewhat slower access.
3. The device’s general heap area is used for images when it is not possible to use the two video RAM areas.
4. Swapping of images to the phone’s flash memory is supported.

Hints for using video memory

The developer should always try to fit commonly used images into the fastest RAM area and use the slower areas for more seldomly used images. This is done by making the MIDlet fetch the commonly used images first and make sure that they fit into the 80 kb of fast video RAM.

To increase the chances that an image to load actually is loaded into fast memory, another image in that area, with at least the same size should be freed. Before allocating the new image into memory, garbage collection (`System.gc()`) should be called.
Another issue to take into consideration when designing applications to use the fastest possible video memory is fragmentation of memory. When an image, allocated between two other images in memory, is freed, only images smaller than the free area can be allocated in that area. Thus, even if the system reports enough free memory for allocation of an image, this may fail, because the free memory consists of several areas, each to small for the image to allocate.

When using very large images, another problem can arise. If an image is too large to fit into the memory dedicated for images, the `Image.createImage()` method may still succeed, because the image is stored in flash memory. However, when the image is to be displayed, it does not fit in the available video memory, and cannot be shown on the screen. The solution is to always estimate the image size in memory before trying to use it in a MIDlet. All images are stored in phone memory in a 16-bit per pixel RGB format, possibly with a 1-bit or 8-bit per pixel alpha-channel. Make sure to save all opaque images with 1-bit alpha, as it draws much faster on the screen.

**Retrieving the IMEI number**

The following command retrieves the IMEI (International Mobile Equipment Identity) number from Sony Ericsson mobile phones:

```
System.getProperty("com.sonyericsson.imei")
```

This returns a string which uniquely identifies a mobile phone, for example: “IMEI 004601-01-501762-8-01” (the exact format of the returned string may differ from the example). Each GSM mobile phone is assigned a unique IMEI code when it is produced. See the following link for further information about IMEI: http://www.numberingplans.com/index.php?goto=guide&topic=imei.

**Note:** The "imei" in the attribute must be written with lowercase letters when the command is used for Sony Ericsson mobile phones, except for the P910 series where uppercase letters must be used instead ("IMEI").

**Network APIs**

Sony Ericsson mobile phones support several network connections:

- HTTP connection
- HTTPS connection (TLS 1.0 is also supported)
  **Note:** HTTPS connections via Proxy is only supported in JP-5.
- Push Registry
- TLS 1.0/SSL 3.0 connections
- Socket connections
• **UDP connections (datagram)**

The following table lists the Network API features and classes of the `javax.microedition.io` package, and their MIDP 2.0 support in Sony Ericsson mobile devices.

<table>
<thead>
<tr>
<th>Feature/Class</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector class</td>
<td>Yes</td>
</tr>
<tr>
<td>All Fields, methods, and inherited methods for the Connector class</td>
<td>Yes</td>
</tr>
<tr>
<td>Mode parameter for the Connector.open() method</td>
<td>No</td>
</tr>
<tr>
<td>The timeouts parameter for the Connector.open() method</td>
<td>No</td>
</tr>
<tr>
<td>HttpConnection interface</td>
<td>Yes</td>
</tr>
<tr>
<td>HttpsConnection interface</td>
<td>Yes</td>
</tr>
<tr>
<td>SecureConnection interface</td>
<td>Yes</td>
</tr>
<tr>
<td>SecurityInfo interface</td>
<td>Yes</td>
</tr>
<tr>
<td>ServerSocketConnection interface</td>
<td>Yes</td>
</tr>
<tr>
<td>UDPDatagramConnection interface</td>
<td>Yes</td>
</tr>
<tr>
<td>PushRegistry class</td>
<td>Yes</td>
</tr>
<tr>
<td>CommConnection interface</td>
<td>No</td>
</tr>
<tr>
<td>Dynamic DNS allocation through DHCP</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Secure sockets and HTTPS connections

HTTPS is supported only for certificates installed on the phone. The following X.509 root certificates for TSL/SSL server authentication are provided by default. However, operators can change which of them are installed and also add other certificates. Local market variations may also exist.

<table>
<thead>
<tr>
<th>Certificate issuer</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verisign</td>
<td>Verisign Class 3 CA</td>
</tr>
<tr>
<td>Baltimore</td>
<td>GTE Cyber Trust Root</td>
</tr>
<tr>
<td>Entrust</td>
<td>Entrust.net Root Certificate</td>
</tr>
<tr>
<td>GlobalSign</td>
<td>GlobalSign Root CA</td>
</tr>
<tr>
<td>Thawte</td>
<td>Thawte Server CA</td>
</tr>
<tr>
<td>RSA data Security</td>
<td>-</td>
</tr>
</tbody>
</table>
When initiating a connection and the certificate cannot be validated, the connection fails and an exception is thrown.

**JAD/manifest attributes**

The application descriptor **must** contain the following attributes:

- MIDlet-Name
- MIDlet-Version
- MIDlet-Vendor
- MIDlet-Jar-URL
- MIDlet-Jar-Size

The application descriptor **may** contain:

- MIDlet-\(<n>\) for each MIDlet
- MIDlet-icon (for the ideal look and feel icon size **16x16 pixels is recommended**) This attribute is only supported in **JP-4** and **JP-5**.
- MicroEdition-Profile (recommended)
- MicroEdition-Configuration (recommended)
- MIDlet-Description
- MIDlet-Data-Size
- MIDlet-Permissions (recommended)
- MIDlet-Permissions-Opt
- MIDlet-Push-\(<n>\)
- MIDlet-Install-Notify
- MIDlet-Delete-Notify
- MIDlet-Delete-Confirm
- MIDlet-Certificate-\(<X>-\(<Y>\)
- MIDlet-Jar-RSA-SHA1
- Any application-specific attributes that do not begin with MIDlet- or MicroEdition-

**Vodafone JAD attributes**

Sony Ericsson mobile phones, manufactured for Vodafone, support the following additional attributes in the JAD/manifest:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDxlet-Resident</td>
<td>Supported values: (Y = \text{Resident MIDlet}, N = \text{Not resident MIDlet}). The attribute value (S = \text{Stay resident}) is not supported.</td>
</tr>
<tr>
<td>MIDxlet-ScreenSize/MIDxlet-Application-Range</td>
<td>Values: (W,H) or (Wmin-Wmax,Hmin-Hmax). Screen size or minimum/maximum width and height expected by the application. Both attributes are supported, but MIDxlet-ScreenSize is recommended. If both attributes are present in a JAD file, MIDxlet-ScreenSize attribute has precedence.</td>
</tr>
</tbody>
</table>
JSR 75 implementation

JSR 75 is only implemented in JP-5. This section specifies which features are supported in these mobile phones.

PIM API

The PIM API supports the following PIM lists:

- Contacts (ContactList)
- Calendar (EventList)
- Tasks (ToDoList)

Actual names of lists and other labels depend on locale.

Contacts

Supported Java PIM fields (native/GUI field names in parenthesis):

- UID (L UID)
- NAME (LastName/Name). Supported array elements:
  - NAME_FAMILY
  - NAME_GIVEN
- ADDR (HomeAddress). Only one address, always ATTR_HOME. Supported array elements:
  - ADDR_STREET (Street)
  - ADDR_LOCALITY (City)
  - ADDR_REGION (State)
  - ADDR_POSTALCODE (Zip code)
  - ADDR_COUNTRY (Country)
- TITLE
- ORG (Company)
- EMAIL
- URL
- NOTE (Freetext)
- TEL, supported attributes (max one + ATTR_PREFERRED):
  - ATTR_HOME (HomeNumber)
  - ATTR_WORK (WorkNumber)
  - ATTR_MOBILE (CellNumber/Mobile number)
  - ATTR_FAX (FaxNumber)
  - ATTR_OTHER (OtherNumber)
  - ATTR_PREFERRED (DefaultNbr), on one number only.
- Numbers are sent to different database containers based on attributes. If two numbers have the same attribute only one is stored. One number with multiple attributes creates copies in different containers (not combined on retrieval). No attribute is treated like ATTR_OTHER. As a consequence of all this, field value indexes are not preserved on retrieval.
- Supported char-set: '0'-'9','*','#','?','+',' and 'p'.
- PHOTO_URL
  - Files that have no Java mapping are not returned on read, for example predefined images that link to system directories.
  - Only local URLs ('file:///') that refer to existing files can be persisted.
Developers Guideline Java J2ME™ for Sony Ericsson mobile phones

- FORMATTED_NAME (LastName/Name)
  - Avoid using this field, since it competes with NAME for the same DB container.
  - To commit, first delete NAME.

Unsupported Java standard fields
- BIRTHDAY
- CLASS
- FORMATTED_ADDR
- NICKNAME
  - Requires vCard 3.0
- PHOTO
- PUBLIC_KEY
- PUBLIC_KEY_STRING
- REVISION

Unsupported native fields
- Birthday
- ChangeCounter
- ContactPosition
- WVID (Presence ID)
- NameVoiceTag (Voice Commands)
- PersonalRing (Ringtone)
- JapaneseReading (Furigana)

Restrictions
- All fields except TEL can have one value only.
- Categories are not supported.
- A maximum of 1000 contacts (2500 phone numbers) can be saved in the phone.

Calendar
Supported Java PIM fields (native/GUI field names in parenthesis):
- UID (LUID).
- SUMMARY (Summary/Description).
- LOCATION (Location).
- NOTE (Description).
- END (EndDateAndTime).
  - Default: current time + 1 second.
- START (StartDateAndTime).
  - Default: current time.
- ALARM (ReminderDateAndTime).
  - Must be positive, that is, before start.
- CLASS (Class).
- REVISION (LastModified)

Unsupported native fields
- TimeZone
- DaylightSaving
- AllDayEvent
Restrictions
- Database must have: ALARM <= START <= END (defaults set on commit).
- RepeatRules are not supported (Recurrence). Only the first item in a recurrence series is retrieved.
- Categories are not supported.
- A maximum of 300 calendar events can be saved in the phone.

Tasks
Supported Java PIM fields (native/GUI field names in parenthesis):

- UID (LUID).
- SUMMARY (Summary/Description).
- NOTE (Description).
- DUE (RemainderDateAndTime/Reminder).
- COMPLETION_DATE (CompletedDateAndTime).
- COMPLETED (Status). Native system currently uses:
  - CAL_STATUS_NOT_STARTED_VALUE (0), mapped to false.
  - CAL_STATUS_IN_PROGRESS_VALUE (1), equated to not started.
  - CAL_STATUS_COMPLETED_VALUE (2), mapped to true.
- PRIORITY (Priority). Native system currently uses:
  - CAL_PRIORITY_HIGH_VALUE (1)
  - CAL_PRIORITY_NORMAL_VALUE (2)
  - CAL_PRIORITY_LOW_VALUE (3)
  - Only 8 bits are persisted (not sign extended on retrieval).
- CLASS (Class).
- REVISION (LastModified).

Unsupported native fields
- TimeZone
- DaylightSaving
- DueDateAndTime

Restrictions
- Categories are not supported.
- A maximum of 80 tasks can be saved in the phone.

Serialization
Serialization includes converting vCards and vCalendar events/todos in serial (text) form into PIM items (FromSerial), and back again (ToSerial).

There are two parsers, one for vCards and one for vCalendar Events/ToDos. As required by the standard, the parsers support vCard 2.1 and vCalendar 1.0, with Quoted-Printable and BASE64 encoding formats. The character encoding must be UTF-8, which means that normal seven bit ASCII is also allowed (since it is a subset of UTF-8).

Only values/properties supported by the databases are copied to the PIM item.

If there are too many values for a particular field, the implementation will favour those with attributes that the field supports.
File Connection API

This section specifies the File Connection API support in JP-5.

The folders listed below and their content (including sub-folders), which are available via the File Manager application in the phone, are available via the FileConnection API as directories (folders) and files. This includes also access to the whole file system on an external memory (removable media), if present.

- `<file:///c/>`
- `<file:///c:/other>`
- `<file:///c:/pictures/>`
- `<file:///c:/sounds/>`
- `<file:///c:/videos/>`
- `<file:///e:/>` (memory card)

**Note:** The folders Games and Themes are not available via the Java File Connection API.

Attempts to access other file areas than the ones specified above, result in a java.lang.SecurityException being thrown to the Java application.

The File Connection API supports the same file/dir attributes as are supported by the built-in File manager application. File and directory names accessed via the File Connection API are case-insensitive.

The length of a file path is limited by the native file system (including the memory card file structure).

**Note:** The Java path is mapped to a native path. The maximum native path is 120 characters.

**Restricted file/directory operations**

The following operations fail if they are performed on any of the built-in roots:

- Create new file in the root directory
- Create new directory in the root directory
- Change attributes
- Delete root or built in directory
- Rename root or built in directory
- Request of last modification date returns 0

**Rules for operations on DRM protected files**

The following operations are supported on DRM protected files:

- Open connection
- List (DRM protected files appear in directory lists)
- Request file size
- Request attributes and last modification date
- Delete
- Exists
- Is directory

The following operations are not supported on DRM protected files:

- Create file
- Change attributes
- Rename file
Truncate file
Open input stream
Open output stream

Rules for operations on Sony Ericsson encrypted files

Sony Ericsson encrypted files are files that are encrypted and stored in the phone memory or on the Memory Stick. These files are not accessible for the user.

The following operations are supported on encrypted files:

- Open connection
- list (encrypted files will appear in directory lists)
- exist
- file size
- can read
- can write
- is hidden
- lastModified
- dirSize (encrypted files are counted)

The following operations are not supported on encrypted files:

- create
- setReadable
- setWritable
- setHidden
- delete
- rename
- truncate
- openInputStream
- openOutputStream
- read / write

Tips for using the JSR 82

Local device

To find out what is supported by the device, use LocalDevice.getProperty(). See JavaDoc for valid properties.
Device discovery

Tip 1
Filter found RemoteDevices immediately by using DeviceClass. By doing this you can avoid unnecessary actions, for example doing a service search on a discovered PC when running a J2ME game.

Tip 2
If retrieving cached remote devices via JSR-82 API, then the information about remote device class is not available. It might be better to implement cache with filtered devices from the initial device discovery.

Tip 3
Only ask for friendly names for the devices displayed in GUI, and save time.
remoteDevice.getFriendlyName(true) is supported in JP-5.

Tip 4
To gain better user experience, present discovered remote devices directly when found. Do not wait until the inquiry is completed.

Games

Use ByteArrayOutputStream/ByteArrayInputStream buffer for RFCOMM.

The transfer rate may be increased by using fixed size byte array, that is, by not having to send the buffer length before sending the actual byte buffer.
Querying system properties

Calls to the Java platform to find out which system properties are supported in a device can be made on different levels, for example what classes are supported in the phone or what properties are supported by a specific class.

Supported classes

To check if a phone supports a specific class, the `Class.forName()` function can be used.

```java
try{
    Class.forName("...");
} catch(Exception ex){
    System.out.println("No support for .....")
}
```

Examples:

```java
Class.forName("javax.microedition.media.Manager"); // JSR135
Class.forName("com.nokia.mid.ui.DeviceControl");  // Nokia UI extension
Class.forName("javax.bluetooth.LocalDevice");  // JSR82
Class.forName("javax.wireless.messaging.MessageConnection");  // JSR120
Class.forName("javax.microedition.pim.PIM");  // JSR75
Class.forName("javax.microedition.m3g.Graphics3D");  // JSR184
Class.forName("com.mascotcapsule.micro3d.v3.Graphics3D");  // Mascotcapsule
```

System.getProperty(String Key) calls

`Java.lang.System.getProperty(String Key)` calls are used to find out what is supported in the phone.

Example:

```java
import java.lang.*;

String value;
String key  = "microedition.pim.version";

value  = System.getProperty( key );
...
```

Standard system properties

The following are examples of standard properties that can be retrieved with the `System.getProperty()` call:
microedition.configuration
microedition.profiles
microedition.encoding
microedition.locale
microedition.platform
microedition.jtwi.version //JSR 185

There is also one Sony Ericsson specific property:

com.sonyericsson.imei

**JSR 120 system properties**

To find out if the API is implemented:

System.getProperty("wireless.messaging.sms.smsc")

**JSR 75 system properties**

To find out what versions of the JSR 75 APIs are implemented in the phone:

System.getProperty("microedition.io.file.FileConnection.version")
System.getProperty("microedition.pim.version")

The following file connection API properties are URLs of default storage directories in the phone, retrieved with the `System.getProperty()` call:

fileconn.dir.photos
fileconn.dir.videos
fileconn.dir.graphics
fileconn.dir.tones
fileconn.dir.music
fileconn.dir.recordings
fileconn.dir.private

Localized names of directories corresponding to the default URLs above are found in the following properties:

fileconn.dir.photos.name
fileconn.dir.videos.name
fileconn.dir.graphics.name
fileconn.dir.tones.name
fileconn.dir.music.name
fileconn.dir.recordings.name
fileconn.dir.private.name

The following call returns localized names to the roots returned by the `FileSystemRegistry.listRoots()` method. The returned names are listed in the same order as returned by this method and are separated by semicolon (;):

System.getProperty("fileconn.dir.roots.names")
**MMAPI system properties**
The following properties can be retrieved from the MMAPI using the `System.getProperty()` call:

- `microedition.media.version`
- `supports.mixing`
- `supports.audio.capture`
- `supports.video.capture`
- `supports.recording`
- `audio.encodings`
- `video.encodings`
- `video.snapshot.encodings`
- `streamable.contents`

To find out which protocols and Content types are supported, the following calls can be made from a `Manager` class object:

```
static java.lang.String[] getSupportedContentTypes(java.lang.string protocol)  // lists supported content types for a given protocol
static java.lang.String[] getSupportedProtocols(java.lang.string content_type)  // lists supported protocols for a given content type
```

From a `Player` class object, a specific Control or the Controls collection supported by the player can be retrieved:

```
Control getControl(java.lang.String ControlType)
Control[] getControls()
```

**Bluetooth Local device properties (JSR 82)**

To find out what Bluetooth API properties are supported in the local device, the `LocalDevice.getProperty("...")` can be called with the following parameters:

- `bluetooth.api.version`
- `bluetooth.master.switch`
- `bluetooth.sd.attr.retrievable.max`
- `bluetooth.connected.devices.max`
- `bluetooth.12cap.receiveMTU.max`
- `bluetooth.sd.trans.max`
- `bluetooth.connected.inquiry.scan`
- `bluetooth.connected.page.scan`
- `bluetooth.connected.inquiry`
- `bluetooth.connected.page`

**Implementation specific properties in JSR 184**

The version of the JSR 184 API is retrieved with:

```
System.getProperty("microedition.m3g.version")
```
Other JSR 184 properties can be retrieved through the `Graphics3D.getProperties("...")` with the following keys:

- `supportAntialiasing`
- `supportTrueColor`
- `supportDithering`
- `supportMipmapping`
- `supportPerspectiveCorrection`
- `supportLocalCameraLighting`
- `maxLights`
- `maxViewportDimension`
- `maxTextureDimension`
- `maxSpriteCropDimension`
- `maxTransformsPerVertex`
- `maxTextureUnits`
Links and references

This chapter contains useful links and references.

Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLDC 1.0 (JSR 30)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=30">http://www.jcp.org/en/jsr/detail?id=30</a></td>
</tr>
<tr>
<td>CLDC 1.1 (JSR 139)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=139">http://www.jcp.org/en/jsr/detail?id=139</a></td>
</tr>
<tr>
<td>JTWI R1 (JSR 185)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=185">http://www.jcp.org/en/jsr/detail?id=185</a></td>
</tr>
<tr>
<td>MIDP 1.0 (JSR 37)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=37">http://www.jcp.org/en/jsr/detail?id=37</a></td>
</tr>
<tr>
<td>MIDP 2.0 (JSR 118)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=118">http://www.jcp.org/en/jsr/detail?id=118</a></td>
</tr>
<tr>
<td>MMAPI (JSR 135)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=135">http://www.jcp.org/en/jsr/detail?id=135</a></td>
</tr>
<tr>
<td>WMA (JSR 120)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=120">http://www.jcp.org/en/jsr/detail?id=120</a></td>
</tr>
<tr>
<td>3D (JSR 184)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=184">http://www.jcp.org/en/jsr/detail?id=184</a></td>
</tr>
<tr>
<td>Bluetooth (JSR 82)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=82">http://www.jcp.org/en/jsr/detail?id=82</a></td>
</tr>
<tr>
<td>Optional Package (JSR 75)</td>
<td><a href="http://www.jcp.org/en/jsr/detail?id=75">http://www.jcp.org/en/jsr/detail?id=75</a></td>
</tr>
<tr>
<td>Mascot Capsule</td>
<td><a href="http://www.mascotcapsule.com">http://www.mascotcapsule.com</a></td>
</tr>
</tbody>
</table>

Java 2 Platform, Micro Edition (J2ME)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sony Ericsson Developer World</td>
<td><a href="http://www.sonyericsson.com/developer/">http://www.sonyericsson.com/developer/</a></td>
</tr>
</tbody>
</table>
Developers Guideline Java J2ME™ for Sony Ericsson mobile phones

<table>
<thead>
<tr>
<th>Feature</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helpful hints (white paper)</td>
<td><a href="http://java.sun.com/j2me/docs/pdf/midpwp.pdf">http://java.sun.com/j2me/docs/pdf/midpwp.pdf</a></td>
</tr>
<tr>
<td>Java Consumer Software Documentation Web site</td>
<td><a href="http://java.sun.com/j2me/docs/">http://java.sun.com/j2me/docs/</a></td>
</tr>
</tbody>
</table>

### 3D developer tools/plugins

<table>
<thead>
<tr>
<th>Feature</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mascot Capsule Micro3D version 4 (JSR 184) plugins</td>
<td><a href="http://www.mascotcapsule.com/M3G/download/e_index.html">http://www.mascotcapsule.com/M3G/download/e_index.html</a></td>
</tr>
</tbody>
</table>
Appendix C
Sony Ericsson J2ME SDK

This appendix contains information about the Sony Ericsson J2ME SDK and its integration in different developers’ tools.
Features

The Sony Ericsson J2ME SDK supports full 3D emulation, Mascot Capsule Ver. 3 and Ver. 4 (JSR 184) are supported. The developer can also take advantage of many WTK features. The SDK includes detailed documentation (JavaDoc) of supported Java APIs (CLDC, MIDP, MMAPI, WMAPI, JSRs).

The SDK also supports On-Device source-level Debugging (ODD). The Sony Ericsson J2ME SDK can be integrated with any UEI compliant Java IDE.

Additionally, the SDK includes useful utilities such as the Device Explorer and ejava.exe command line tool. These provide an interface for manipulating the device’s application manager. The developer can install, remove, start, stop, pause, and resume Java applications. The Device Explorer also provides an interface for displaying heap and file system statistics, requesting garbage collection to run, enabling KVM trace messages, and enabling serial network emulation.

Before installing the Sony Ericsson J2ME SDK, you will need to have the SDK for Java 2, Standard Edition (version 1.4.2 is recommended), installed. If you wish to install an IDE, you may do so either before or after installing the Sony Ericsson J2ME SDK. Note that an IDE is not required but is highly recommended.

The latest version of the Sony Ericsson J2ME SDK is available for download at www.sonyericsson.com/developer/java. The Sony Ericsson J2ME SDK is required for ODD.

Sony Ericsson J2ME SDK integration in JBuilder 2005

In the instructions below, <SDK_Path> stands for the path where the Sony Ericsson J2ME SDK is installed.

To add Sony Ericsson Emulators:
1. In the Jbuilder window, select the menu Tools/Configure/JDKs…
2. Choose New
3. Click on Existing JDK home path and browse to <SDK_Path>/PC_Emulation/WTK1
4. Click OK to add the Sony Ericsson WTK1 emulators
5. Repeat steps 2 to 4 and add the WTK2 and the On-Device Debug emulators. The paths to use in step 3 are:
   - <SDK_Path>/PC_Emulation/WTK1
   - <SDK_Path>/PC_Emulation/OnDeviceDebug

To set which device will be used for a Specific JDK:
1. In the Jbuilder window, select the menu Tools/Configure/JDKs…
2. Chose one of the JDK: s added above
3. Click the Micro tab
4. Choose the preferred device in the Target Device drop-down list

**To use Sony Ericsson emulators in a project:**
1. Right-click on the project icon and chose Properties... or choose the menu item Project/Project Properties
2. Select Paths in the upper left
3. Click on the “…” button at JDK: and select the emulator you want to use

**To use Sony Ericsson On Device Debug in a project:**
1. Make sure the project has been built so that there is a valid JAD file
2. Follow the steps in To use Sony Ericsson Emulators in a Project above and choose the On Device Debug emulators
3. Choose Run/Configurations…
4. Chose Edit on the selected configuration
5. Select the radio button JAD file, browse and select the project’s JAD file
6. Click OK

**Sony Ericsson J2ME SDK integration in NetBeans 4**

In the instructions below, <SDK_Path> stands for the path where the Sony Ericsson J2ME SDK is installed.

**To add Sony Ericsson emulators:**
1. In the Netbeans 4 window, select the menu Tools/Java Platform Manager (or File/<project name> properties... and click on Manage Emulators...)
2. Select J2ME and click on Add Platform...
3. Browse to <SDK_Path>/PC_Eulation/WTK1
4. Click Finish to add the Sony Ericsson WTK1 emulators
5. Repeat steps 2 to 4 and add the WTK2 and the On Device Debug emulators. The paths to use in step 3 are:
   - <SDK_Path>/PC_Eulation/WTK2
   - <SDK_Path>/PC_Eulation/OnDeviceDebug

**To set the platform and device that will be used for emulation:**
1. In the Netbeans 4 window, select the menu File/<project name> properties... and select Platform in the table to the left.
2. Use the drop-down list Project Configuration to choose one of the platforms added above.
3. Use the drop-down list *Device* to choose one of the emulators available on the chosen platform.

4. Click OK

**Note:** If NetBeans complains about the missing file "zayit.dll" when trying to run your project with a Sony Ericsson emulator, you should reboot the computer and try again.

**To use Sony Ericsson On Device Debug in a project:**
In the Netbeans 4 window, select the menu *Run/Debug Main Project*, or press F5. Note that you must choose *Debug Main Project*, not *Run Main Project*. 
## Index

### Numerics
- 3D API ................................................................. 15

### A
- abbreviations .................................................. 4
- audio support .................................................. 14

### C
- CLDC .................................................................. 26
- command types .................................................. 17

### D
- debug interface .................................................. 27

### E
- error messages .................................................. 18

### J
- JTWI ................................................................. 26

### L
- links and references ........................................... 48

### M
- memory ............................................................. 16
- MIDP ................................................................. 12
- MMAPI ............................................................. 13, 26

### N
- navigation key ................................................... 17

### P
- port numbers .................................................... 12

### V
- video support .................................................... 14

### W
- WMA ............................................................... 12, 26