



Using MIDP

MIDP Reference Implementation, Version 2.0 FCS

Java™ 2 Platform, Micro Edition

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, California 95054
U.S.A. 650-960-1300

November, 2002

Copyright © 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Sun, Sun Microsystems, the Sun logo, Solaris, Java, J2ME, J2SE, Java Developer Connection, Forte, Javadoc, and Java Powered logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

The Adobe® logo is a registered trademark of Adobe Systems, Incorporated.

Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats - Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y ena.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Sun, Sun Microsystems, le logo Sun, Solaris, Java, J2ME, J2SE, Java Developer Connection, Java™ 2 runtime environment, Forte, Javadoc, et Java Powered logo sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

UNIX est une marque enregistree aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Le logo Adobe® est une marque déposée de Adobe Systems, Incorporated.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please
Recycle



Adobe PostScript

Contents

Preface xi

1. Using the Graphical User Interface 3

Overview of the Device Emulator 4

Starting the Emulator 6

Downloading and Installing a MIDlet Suite 7

 Common Download and Installation Steps 7

 Additional Steps for Push Functionality 12

Running a MIDlet or MIDlet Suite 14

 Launching a MIDlet or MIDlet Suite 14

 Handling Permission Requests 15

Getting Information on a MIDlet Suite 16

Removing a MIDlet Suite 17

Updating a MIDlet Suite 19

Changing a MIDlet Suite's Permission Levels 20

Getting Information About MIDP 23

2. Using the `midp` Executable 25

General Instructions 26

Improving Device Simulation 26

 Changing a Property Value 27

 Common Property Values to Update 27

Installing a MIDlet Suite	29
Listing Installed MIDlet Suites	30
Removing an Installed MIDlet Suite	31
3. Using MIDP Security Features	33
Overview	33
Trusted and Untrusted MIDlets	33
Permissions	35
Protection Domains	36
The Authorization Process	38
Managing the Security Policy	38
4. Managing Public Keys of Certificate Authorities	41
Overview	41
General Instructions for MEKeyTool	42
Working With Multiple ME Keystores	43
Creating Alternate ME Keystores	43
Managing Alternate ME Keystores	43
Running MIDP and Alternate Keystores	44
Importing a Key	44
Listing Available Keys	45
Deleting a Key	46
Replacing a Key	46
Handling Certificate Exceptions When Running MIDP	47
A. The JadTool Utility	49
Synopsis	49
Description	49
Options	50
Examples	52
See Also	52

B. The `MEKeyTool` Utility 53

Synopsis 53

Description 53

Options 54

Examples 55

See Also 55

C. The `midp` Command 57

Synopsis 57

Description 57

Options 61

Examples 63

See Also 64

D. The `preverify` Tool 65

Synopsis 65

Description 65

Options 66

Operands 67

Exit Status 67

Examples 68

Environment Variables 69

See Also 69

Index 71

Figures

FIGURE 1	Phone Skin	4
FIGURE 2	Navigation Buttons on the Phone Skin	5
FIGURE 3	Other Buttons Above the Phone Keypad on the Phone Skin	5
FIGURE 4	Standard Phone Keypad on the Phone Skin	5
FIGURE 5	Power Button on the Phone Skin	6
FIGURE 6	Welcome Screen	6
FIGURE 7	Applications Screen With No Installed MIDlets	7
FIGURE 8	System menu for the Install Application MIDlet	8
FIGURE 9	MIDlet Locator Screen	8
FIGURE 10	MIDlet Locator Screen With a URL Entered	9
FIGURE 11	System menu for the MIDlet Locator	9
FIGURE 12	List of Available MIDlets	10
FIGURE 13	Confirmation Screen for Installing a MIDlet Suite	10
FIGURE 14	Applications Screen	11
FIGURE 15	Installation Stopping in Response to Being Denied a Required Permission	12
FIGURE 16	Request to use the Push Functionality	12
FIGURE 17	Request to Receive Incoming Data	13
FIGURE 18	MIDlet Being Launched	14
FIGURE 19	Request for Permission to Use Protected Functionality	15
FIGURE 20	System menu for the Applications Screen	16
FIGURE 21	Information Screen for a MIDlet Suite	17

FIGURE 22	System menu for the Applications Screen	18
FIGURE 23	Confirmation Screen for Removing a MIDlet Suite	18
FIGURE 24	System menu for the Applications Screen	19
FIGURE 25	Confirmation Screen for Updating a MIDlet Suite	20
FIGURE 26	System menu for the Applications Screen	21
FIGURE 27	Application Settings for a MIDlet Suite	21
FIGURE 28	Confirmation of Saved Settings	22
FIGURE 29	System menu for the Applications Screen	23
FIGURE 30	About Box for the MIDP Reference Implementation	24
FIGURE 31	Example Certificate Chain	34

Tables

TABLE 1	Permission Names Defined in the <i>MIDP 2.0 Specification</i>	35
TABLE 2	Properties in the <code>internal.config</code> File	58
TABLE 3	Properties in the <code>system.config</code> File	60

Preface

The *Using MIDP* describes how to manage the device emulator and MIDlet suites. It covers setting the security policy for the MIDP Reference Implementation's device emulator, managing the public keys available to the emulator, and using the emulator to run and manage MIDlets and MIDlet suites. It assumes that you have already installed the product, as described in *Installing MIDP*.

How This Book Is Organized

This book has the following chapters and appendices:

[Chapter 1](#) shows the device emulator's graphical user interface and describes how to interact with it to manage and run MIDlet suites.

[Chapter 2](#) shows you how to use the command-line interface to the device emulator to manage and run MIDlet suites.

[Chapter 3](#) describes the MIDP Reference Implementation's security features and shows you how to manage the security policy of the device emulator.

[Chapter 4](#) shows you how to manage the certificate authority keys needed for secure network protocols and MIDlet authorization.

[Appendix A](#) describes the `JadTool` utility using a man-page format.

[Appendix B](#) describes the `MEKeyTool` utility using a man-page format.

[Appendix C](#) describes the `midp` command using a man-page format.

[Appendix D](#) describes the `preverify` tool using a man-page format.

Using Operating System Commands

This document may not contain information on basic UNIX[®] or Microsoft Windows commands and procedures such as opening a terminal window, changing directories, and setting environment variables. See the software documentation that you received with your system for this information.

Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% su Password:
<i>AaBbCc123</i>	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type <code>rm filename</code> .

Shell Prompts

Shell	Prompt
C shell	%
Microsoft Windows	<i>directory></i>

Related Documentation

The following documentation is included with this release:

Application	Title
All	<i>Release Notes</i>
Installing	<i>Installing MIDP</i>
Running and managing security for emulator	<i>Using MIDP</i>
Porting the MIDP Reference Implementation	<i>Porting MIDP</i>
Creating and building MIDlets	<i>Creating MIDlet Suites</i>
Viewing reference documentation created by the Javadoc™ tool	<i>API Reference</i>
Looking at examples	<i>Example Overview</i>

Accessing Sun Documentation Online

The Java Developer Connectionsm web site enables you to access Java™ platform technical documentation on the Web:

<http://developer.java.sun.com/developer/infodocs/>

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

docs@java.sun.com

Using the Graphical User Interface

The MIDP Reference Implementation runs MIDlets in a device emulator, which shows how the MIDlets will look and feel on a device. This chapter shows you how to start the device emulator's graphical user interface (GUI), and how to interact with it to run and manage MIDlets. The chapter contains the sections:

- [Overview of the Device Emulator](#)
- [Starting the Emulator](#)
- [Downloading and Installing a MIDlet Suite](#)
- [Running a MIDlet or MIDlet Suite](#)
- [Getting Information on a MIDlet Suite](#)
- [Removing a MIDlet Suite](#)
- [Updating a MIDlet Suite](#)
- [Changing a MIDlet Suite's Permission Levels](#)
- [Getting Information About MIDP](#)

Overview of the Device Emulator

The device emulator is a cellular phone. The emulator's GUI displays a *skin*, which is a graphic that looks like a device. The skin has a screen and keypad. The following figure shows the skin:



FIGURE 1 Phone Skin

You press the buttons on the device by clicking them with the left mouse button (left-clicking) or by using their keyboard shortcuts. The buttons and keyboard shortcuts are described below.

The skin has up, down, left, and right navigation buttons, and a select button in the center of the navigation buttons as shown in [FIGURE 2](#). You can use the arrow keys on the keyboard of your desktop system instead of the navigation buttons, and the Enter key for the Select button.

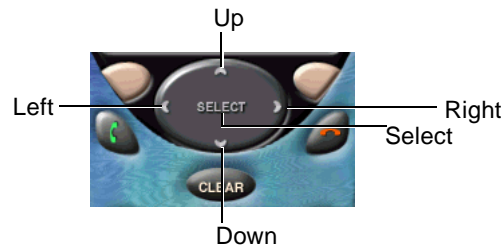


FIGURE 2 Navigation Buttons on the Phone Skin

The skin also has a Start Call button (the button on the left printed with a green handset), an End Call button (the button on the right printed with a red handset), a Clear button, and two soft buttons to either side of the up navigation button. [FIGURE 3](#) shows these buttons. You can use the F1 and F2 keys on the keyboard of your desktop system instead of the left and right soft buttons. You can use the F10 key instead of the End Call button. When you are entering text, you can use the backspace key instead of the Clear button.



FIGURE 3 Other Buttons Above the Phone Keypad on the Phone Skin

The skin has a standard phone keypad: the numbers 0-9, along with the pound (#) and star (*) function keys. The mode-shift and space functions are mapped to the * and # keys, respectively, on the keypad. You can use the letter, number, #, *, shift, and space keys on your keyboard instead.



FIGURE 4 Standard Phone Keypad on the Phone Skin

Finally, the skin has a power button above the screen, as shown in the following figure, which exits the emulator. Closing the emulator window also exits it.

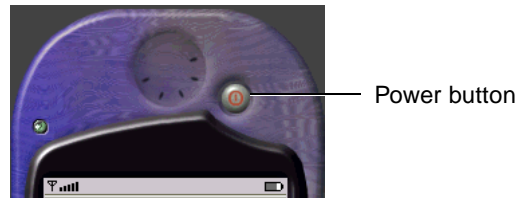


FIGURE 5 Power Button on the Phone Skin

Starting the Emulator

1. Open a terminal window.
2. Change your current directory to the directory that holds your installation of the MIDP Reference Implementation.

For example, if your installation is in the `c:\midp2.0fcs` directory, you could execute the following command:

```
c:\> cd midp2.0fcs
```

3. Run the `midp` command with no arguments.

See [Appendix C, "The midp Command"](#) for more information in a manpage format. For example, you could use the following command:

```
c:\midp2.0fcs> bin\midp
```

The device skin will appear displaying the Java Powered logo™ welcome screen.



FIGURE 6 Welcome Screen

Downloading and Installing a MIDlet Suite

This section describes both how to install a MIDlet suite, and the additional steps that occur if the MIDlet suite uses push functionality. This section has the topics:

- [Common Download and Installation Steps](#)
- [Additional Steps for Push Functionality](#)

Common Download and Installation Steps

To download and install a MIDlet suite:

1. Start the Emulator as described in [“Starting the Emulator” on page 6](#).
2. Press the button under the Apps label on the device skin to go to the Applications Screen.

An Applications screen similar to the one in the following figure will appear:

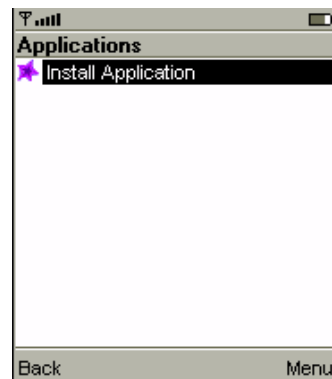


FIGURE 7 Applications Screen With No Installed MIDlets

3. If necessary, move the highlight to the Install Application element.

4. Launch the application installation MIDlet:

- a. Press the button under the Menu label on the device skin.**

A system menu similar to the one in the following figure will appear:

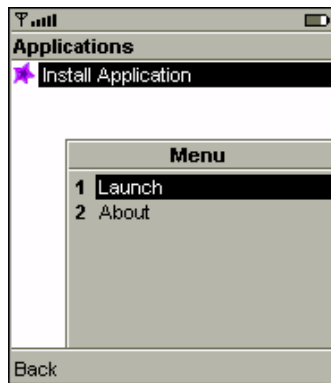


FIGURE 8 System menu for the Install Application MIDlet

- b. Move the highlight to the Launch element in the menu by using the navigation buttons shown in [FIGURE 2](#).**
- c. Press Select.**

A MIDlet Locator screen similar to the following figure will appear:



FIGURE 9 MIDlet Locator Screen

5. Enter the URL of an HTML page that has links to one or more JAD files for MIDlet suites.

After you enter a URL the MIDlet Locator screen will be similar to the one in the following figure:



FIGURE 10 MIDlet Locator Screen With a URL Entered

6. Choose Go:

- a. Press the button under the Menu label on the device skin.

A system menu similar to the one in the following figure will appear:

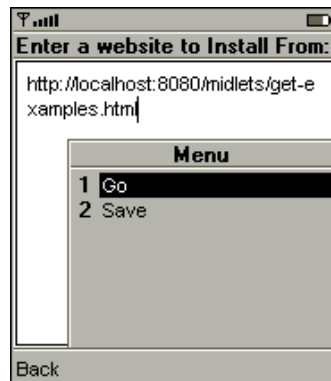


FIGURE 11 System menu for the MIDlet Locator

- b. Move the highlight to the Go element in the menu by using the navigation buttons shown in [FIGURE 2](#).

c. **Press Select.**

A list of the MIDlet suites available on the HTML page will appear. The screen showing the list of MIDlets will be similar to the one in the following figure:

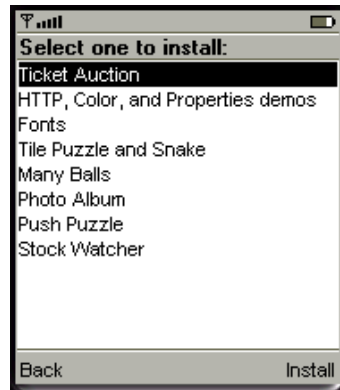


FIGURE 12 List of Available MIDlets

7. **Move the highlight to the MIDlet suite you want to install.**

8. **Choose Install by pressing the button under the Install label on the device skin.**

A confirmation screen will appear, similar to the one shown in the following figure:



FIGURE 13 Confirmation Screen for Installing a MIDlet Suite

9. Confirm by pressing the button under the Install label on the device skin.

Cancel by pressing the button under the Cancel label on the device skin.

If you chose Install, MIDP will try to download and install the MIDlet suite. A series of screens will inform you of the progress of the installation, including informing you if there is a problem with the suite that prevents it from being installed.

If the installation is successful you will be returned to the Application screen and the newly installed MIDlet suite will be highlighted. If the installation fails, you will be given an error message and returned to the MIDlets list in the Install application. The following figure shows the result of a successful installation:

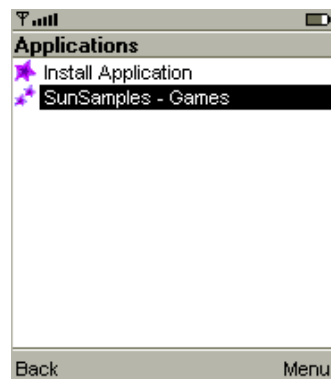


FIGURE 14 Applications Screen

Additional Steps for Push Functionality

Push functionality enables MIDP to launch a MIDlet so that it can receive an incoming message. When the emulator assigns a MIDlet suite that uses the push functionality to a domain that gives push a user permission, you must grant that permission before the emulator can install the MIDlet suite. (See [Chapter 3, “Using MIDP Security Features”](#) for more information on domains and permissions.) If you deny a permission request, a screen similar to the one in the following figure appears:

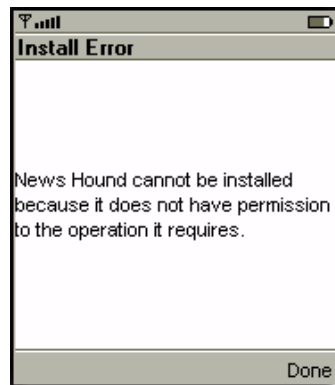


FIGURE 15 Installation Stopping in Response to Being Denied a Required Permission

If you give permission and then change your mind, you can modify your selection. See [“Changing a MIDlet Suite’s Permission Levels”](#) on page 20 for instructions.

The requests for permission occur after you confirm that you want to install the MIDlet ([Step 9](#), above). You must answer the questions before the installation can be completed. The requests are for:

- Permission to use the push functionality

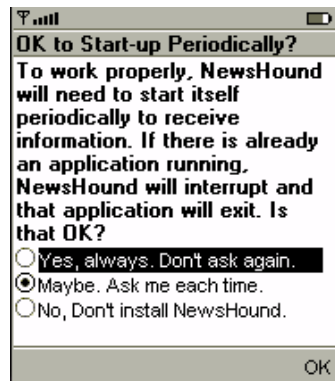


FIGURE 16 Request to use the Push Functionality

- Request to receive messages:

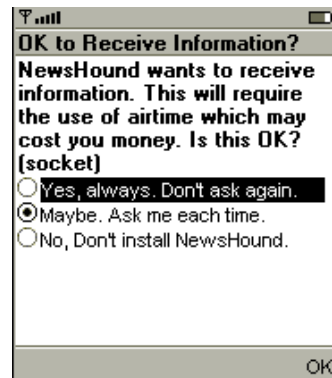
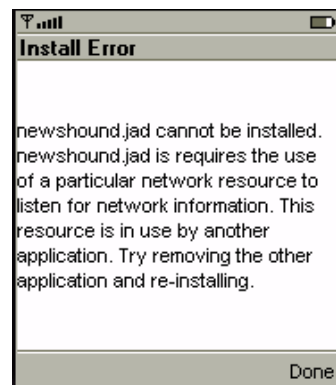


FIGURE 17 Request to Receive Incoming Data

You cannot have multiple MIDlets on the device emulator use the same port. If, after you give any permissions required, the device discovers that the MIDlet suite uses the same port as an installed MIDlet suite, the installation will fail. The screen that informs you of the failure will look similar to the one in the following figure:



Running a MIDlet or MIDlet Suite

This section assumes that you have installed one or more MIDlets suites on the device emulator. There are instructions in the previous section, [“Downloading and Installing a MIDlet Suite” on page 7](#) and in [“Installing a MIDlet Suite” on page 29](#). It contains the topics:

- [Launching a MIDlet or MIDlet Suite](#)
- [Handling Permission Requests](#)

Launching a MIDlet or MIDlet Suite

This section is titled “Launching a MIDlet or MIDlet Suite” because the MIDP Reference Implementation presents both. That is, MIDlet suites can contain one or more MIDlets. In the MIDP Reference Implementation, if a MIDlet suite has only one MIDlet, the device emulator presents that MIDlet and you can choose to run it. If a MIDlet suite has multiple MIDlets, the device emulator presents the MIDlet suite and if you choose to run it, you must then choose which MIDlet to run.

To run a MIDlet or MIDlet suite installed on the device emulator:

1. **Start the Emulator** as described in [“Starting the Emulator” on page 6](#).
2. **Move the highlight to the MIDlet or MIDlet suite you want to run.**
3. **Press Select.**

The MIDlet or MIDlet suite then starts. The following figure shows the TicketLand MIDlet starting:

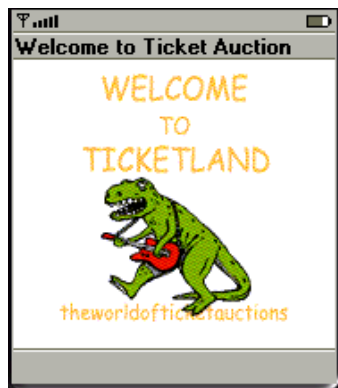


FIGURE 18 MIDlet Being Launched

Handling Permission Requests

When a MIDlet tries to use protected functionality for which it requires user permission, you will be asked to grant permission. A permission screen will appear that looks similar to the one shown in the following figure:

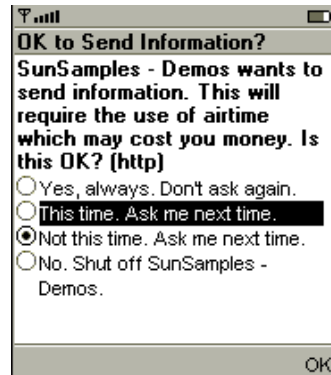


FIGURE 19 Request for Permission to Use Protected Functionality

Note that the protocol for which the permission is required is shown in square brackets after the question. This is for development purposes, not for end users.

In addition to the question in [FIGURE 19](#), the MIDP Reference Implementation also asks, when necessary, for permission to interrupt, to receive information, and to make a network connection. Although the question will vary depending on which protected functionality the MIDlet is trying to use, the possible answers stay the same. The possible answers are:

- Yes, Always. Don't ask again.

Any time this MIDlet wants to use the network it may, without interrupting to ask permission. Selecting this option means that not only can the MIDlet use the network until you exit the MIDlet, but that it can use the network every other time you run the MIDlet too (or for push, every other time it needs to interrupt).

- This time. Ask me again next time.

The MIDlet may perform this one protected action. The next time the MIDlet tries to perform the action, the screen will appear again.

- Not this time. Ask me next time.

The MIDlet may not perform this protected action, but you might allow the MIDlet to do so next time. The next time the MIDlet tries to perform the action, the screen will appear again.

- No. Shut off *MidletName*.

This MIDlet may not perform the protected action. The MIDlet will still appear in the Applications screen, but if you run it, it will not be permitted to use that protected functionality. For example, if the NewsHound MIDlet is shut off, you cannot access any articles or get new ones.

If you change your mind about the permission you have granted, you can update your choice. See [“Changing a MIDlet Suite’s Permission Levels” on page 20](#) for instructions.

Getting Information on a MIDlet Suite

This section assumes that you have installed one or more MIDlet suites on the device emulator. There are instructions in the previous section, [“Downloading and Installing a MIDlet Suite” on page 7](#) and in [“Installing a MIDlet Suite” on page 29](#).

To get information about a MIDlet suite installed on the device emulator:

1. Start the Emulator as described in [“Starting the Emulator” on page 6](#).
2. Move the highlight to the MIDlet suite about which you want information.
3. Choose Info:
 - a. Press the button under the Menu label on the device skin.

A system menu will appear that looks like the one in the following figure:

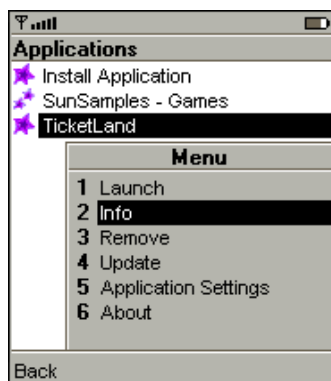


FIGURE 20 System menu for the Applications Screen

- b. Move the highlight to the Info element in the menu by using the navigation buttons shown in [FIGURE 2](#).

c. **Press Select.**

A screen appears that is similar to the one in the following figure:



FIGURE 21 Information Screen for a MIDlet Suite

The arrow at the bottom of the screen indicates that you can see more information by scrolling. Scroll by pressing the down navigation button. (FIGURE 2 shows the navigation buttons.)

4. **Return to the Applications screen by pressing the button under the Back label on the device skin.**

Removing a MIDlet Suite

This section assumes that you have installed one or more MIDlet suites on the device emulator. There are instructions in the previous section, [“Downloading and Installing a MIDlet Suite” on page 7](#) and in [“Installing a MIDlet Suite” on page 29](#).

When you remove a MIDlet suite from a device, you remove not only the suite but also any data that its MIDlets stored on the device. You must always remove a MIDlet suite; you cannot remove a MIDlet from a MIDlet suite.

To remove a MIDlet suite installed on the device emulator:

1. **Start the Emulator as described in [“Starting the Emulator” on page 6](#).**
2. **Move the highlight to the MIDlet suite you want to remove.**

3. Choose Remove:

- a. Press the button under the Menu label on the device skin.**

A system menu like the one in the following figure will appear:

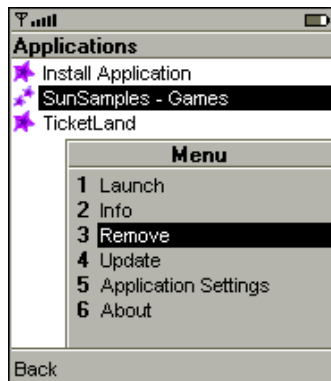


FIGURE 22 System menu for the Applications Screen

- b. Move the highlight to the Remove element in the menu by using the navigation buttons shown in [FIGURE 2](#).**
- c. Press Select.**

A confirmation screen appears that is similar to the one in the following figure:

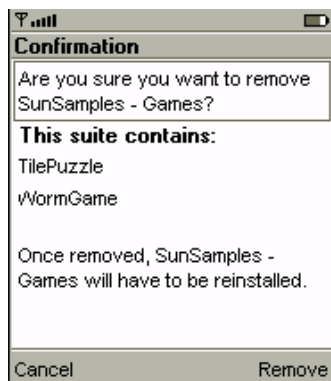


FIGURE 23 Confirmation Screen for Removing a MIDlet Suite

4. **Confirm by pressing the button under the Remove label on the device skin.**

Cancel by pressing the button under the Cancel label on the device skin.

An Applications screen will appear. If you chose Confirm, the MIDlet you removed will have been removed from the list of available applications.

Updating a MIDlet Suite

This section assumes that you have installed one or more MIDlet suites on the device emulator. There are instructions in the previous section, [“Downloading and Installing a MIDlet Suite” on page 7](#) and in [“Installing a MIDlet Suite” on page 29](#).

When you update a MIDlet suite, you replace the resources (the MIDlets, graphics, and so on) but you do not affect any data that its MIDlets stored on the device. You must always update a MIDlet suite; you cannot update a MIDlet from a MIDlet suite.

To update a MIDlet suite installed on the device emulator:

1. **Start the Emulator as described in [“Starting the Emulator” on page 6](#).**
2. **Move the highlight to the MIDlet suite you want to update.**
3. **Choose Update:**
 - a. **Press the button under the Menu label on the device skin.**

A system menu like the one in the following figure will appear:

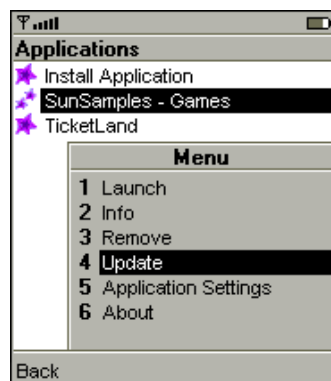


FIGURE 24 System menu for the Applications Screen

- b. **Move the highlight to the Update element in the menu by using the navigation buttons shown in [FIGURE 2](#).**

c. Press Select.

The system then begins to update the MIDlet suite. It uses the same URL from which you originally downloaded and installed the MIDlet suite. A confirmation screen appears, similar to the one in the following figure:



FIGURE 25 Confirmation Screen for Updating a MIDlet Suite

The message that you receive depends on whether the version found is newer, older, or the same as your current version.

4. Confirm by pressing the button under the Continue label on the device skin.

Cancel by pressing the button under the Cancel label on the device skin.

If you chose Continue, the version of the MIDlet suite found at the URL will replace the version of the MIDlet suite currently installed, and you will be returned to the Applications screen.

Changing a MIDlet Suite's Permission Levels

This section assumes that you have installed one or more MIDlet suites on the device emulator. There are instructions in the previous section, [“Downloading and Installing a MIDlet Suite” on page 7](#) and in [“Installing a MIDlet Suite” on page 29](#).

You can change the permission levels that you have granted a MIDlet suite. For example, if you have turned off a permission, you can turn it back on at any level. For example, you could have the MIDlet suite's MIDlets ask permission each time one of them wants to use the protected functionality. See [“Handling Permission Requests” on page 15](#) for a list of the protected functionalities for which you might be asked to set a permission, and the available permission levels.

To change a MIDlet suite's permissions:

1. Start the Emulator as described in [“Starting the Emulator” on page 6](#).
2. Move the highlight to the MIDlet suite for which you want to update permissions.
3. Choose Application Settings:

a. Press the button under the Menu label on the device skin.

A system menu like the one in the following figure will appear:

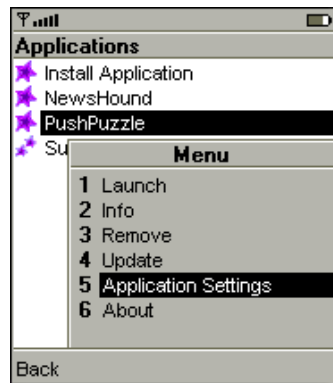


FIGURE 26 System menu for the Applications Screen

- b. Move the highlight to the Application Settings element in the menu by using the navigation buttons shown in [FIGURE 2](#).
- c. Press Select.

A screen will appear that is similar to the one in the following figure. The figure has many parts; each part shows part of the screen, so that you can see all of the questions.



FIGURE 27 Application Settings for a MIDlet Suite

4. Choose the application settings you want.

The Application Settings screen will include all the permissions that you are permitted to set for the suite. The permission levels are similar to the answers given for the permission requests in [“Handling Permission Requests” on page 15](#).

5. Save your choices by pressing the button under the Save label on the device skin.

Cancel by pressing the button under the Back label on the device skin.

In both cases, you will be returned to the Applications screen. If you choose Save, your return will be preceded by a confirmation screen similar to the one in the following figure:



FIGURE 28 Confirmation of Saved Settings

Getting Information About MIDP

The MIDP Reference Implementation includes an About Box for MIDP. The information includes the release number and date.

To see the about box:

1. Start the Emulator as described in [“Starting the Emulator” on page 6](#).
2. Choose About:
 - a. Press the button under the Menu label on the device skin.

A system menu like the one in the following figure will appear:

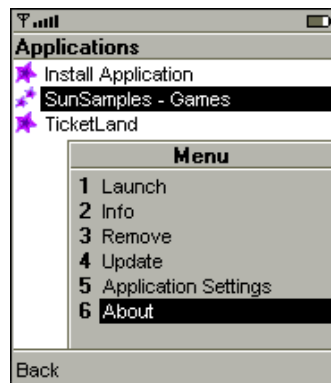


FIGURE 29 System menu for the Applications Screen

- b. Move the highlight to the About element in the menu by using the navigation buttons shown in [FIGURE 2](#).

The About choice is available from the system menu associated with any MIDlet or MIDlet suite on the Applications screen, including the Install Application MIDlet.

c. Press Select.

A screen similar to the one in the following figure appears:



FIGURE 30 About Box for the MIDP Reference Implementation

- 3. Return to the Applications screen by pressing the button under the Done label on the device skin.**

Using the midp Executable

The `midp` command runs a device emulator on your desktop system. You can control many of its runtime characteristics, such as the number of colors that it uses, to better simulate how MIDP will behave when it runs on a device.

This chapter shows you how to use the `midp` command in the sections:

- [General Instructions](#)
- [Improving Device Simulation](#)
- [Installing a MIDlet Suite](#)
- [Listing Installed MIDlet Suites](#)
- [Removing an Installed MIDlet Suite](#)

See [Appendix C, “The midp Command”](#) for detailed information in a manpage format.

General Instructions

The `midp` command is in the `midpInstallDir\bin` directory. To use the command:

1. **Open a command prompt or terminal window.**

2. **Change your current directory to `midpInstallDir`.**

For example, if MIDP Reference Implementation was installed in the directory

```
c:\midp2.0fcs:
```

```
c:\> cd midp2.0fcs
```

3. **Run the `midp` command with any commands or options.**

The commands and options to the `midp` command are described below. Using no commands or options starts the emulator and displays the default device skin. See [Chapter 1, “Using the Graphical User Interface”](#) for more information. The `-help` option provides usage information. For example, you could get help with the command:

```
c:\midp2.0fcs> bin\midp -help
```

The next sections describe some common options.

Improving Device Simulation

The MIDP Reference Implementation uses properties to help the MIDP runtime better simulate the capabilities of a device. For example, by default the MIDP runtime supports multiple color depths. Developers porting MIDP to a device that supports only gray scale can change the value of a property so that the emulator also uses gray scale.

The MIDP configuration properties are stored in two configuration files:

- `midpInstallDir\build\share\lib\internal.config`
- `midpInstallDir\build\share\lib\system.config`

See [Appendix C, “The `midp` Command”](#) for a complete list of the properties in each file. This section covers two topics:

- [Changing a Property Value](#)
- [Common Property Values to Update](#)

Changing a Property Value

To permanently change a property value, edit the appropriate file. The files have each property name-value pair on a single line. The name and value are separated by a colon, as shown below:

```
propertyName : propertyValue
```

The whitespace around the colon is not significant.

To temporarily change a property value, use the `-D` command-line option when you run MIDP Reference Implementation. The option has the following syntax:

```
-DpropertyName=propertyValue
```

For example, to have the emulator temporarily use four colors instead of the default 256 colors, you would change the value of the `system.display.screen_depth` property to 2. You could enter the following command:

```
c:\midp2.0fcs> bin\midp -Dsystem.display.screen_depth=2
```

You can adjust any number of properties from the command line. For example, if you wanted to run installed MIDlet suite number one, simulating a device that had four colors and did not have double-buffering, you could enter the following command:

```
c:\midp2.0fcs> bin\midp -run 1 -Dsystem.display.screen_depth=2  
-Dsystem.display.double_buffered=false
```

Common Property Values to Update

It is common to have to update the following property values:

- `com.sun.midp.io.http.proxy` – Set this property if you must go through a proxy server when you use HTTP to reach the internet. The MIDP implementation will delegate the device emulator's HTTP requests to the proxy. The value of this property has the form *hostName:portNumber*. For example:

```
c:\midp2.0fcs> bin\midp  
-Dcom.sun.midp.io.http.proxy=TheProxy:8080
```
- `com.sun.midp.midlet.platformRequestCommand` – Set this property if you are running a MIDlet that calls the MIDlet class's `platformRequest` method. Its value should be a command that takes a URL as an argument, such as a browser, or a script, and properly handles the URL. If the URL is for a JAD file, the command must install the named package; if the URL is of the form `tel:number`, as specified in RFC2806 (<http://rfc.net/rfc2806.html>), the command must, if it is possible on the platform, make a voice call to *number*.

The following code example shows this property being set to use the Mozilla browser on a machine that has installed the browser in its default location:

```
c:\midp2.0fcs> bin\midp
-Dcom.sun.midp.midlet.platformRequestCommand="C:\Program
Files\mozilla.org\Mozilla\mozilla.exe"
```

If you set this property incorrectly, one of two errors will be displayed in the terminal window where you called the midp command:

- PlatformRequest is not configured.

This happens when the property has no value.

- Spawning a handler process failed. Check the platformRequest configuration.

This happens if the property value is incorrect or if there has been a very rare and unexpected system error.

- com.sun.midp.io.http.force_non_persistent – Set this property to true if you see problems with persistent connections when accessing your web server. These problems could include incorrectly indicating the server sent bad data, displaying only parts of documents and images, and other problems. (You could also try turning off persistent-request capabilities in your web server.) The following code example shows this property being set to true:

```
c:\midp2.0fcs> bin\midp
-Dcom.sun.midp.io.http.force_non_persistent=true
```

- system.i18n.encoding and system.i18n.lang – Set these properties if your device will use something other than a Western European encoding (latin1) for the English language. The following code example shows these properties being set to support a Japanese encoding and language:

```
c:\midp2.0fcs> bin\midp -Dsystem.i18n.encoding=ISO2022_JP
-Dsystem.i18n.lang=ja
```

Installing a MIDlet Suite

A MIDlet suite that is both properly packaged and accessible to a web server can be installed on the emulator. It must be accessible to a web server so that the emulator can download it. The URL from which to download and install a MIDlet should be the location of the MIDlet suite's JAD file. The JAD file contains the URL of the MIDlet suite's JAR file; that file must also be accessible to the web server.

For example, if the JAD file for the example TicketLand MIDlet suite were available at `http://localhost:8080/j2me/midp/games.jad`, the following example shows a command that would install it:

```
c:\midp2.0fcs> bin\midp -install
http://localhost:8080/j2me/midp/auction.jad
Storage name:
#Sun%0020#Microsystems%002c%0020#Inc%002e_#Ticket%0020#Auction_
```

If the MIDlet suite is unsigned it will, by default, be put in the untrusted domain. (See [“Associating MIDlet Suites and Protection Domains” on page 37](#) for more information.) You can change its domain by using the `-domain` option. The following example shows the unsigned TicketLand MIDlet suite being installed in the trusted domain:

```
c:\midp2.0fcs> bin\midp -install -domain trusted
http://localhost:8080/j2me/midp/auction.jad
Storage name:
#Sun%0020#Microsystems%002c%0020#Inc%002e_#Ticket%0020#Auction_
```

If you use the `-install` option to install a MIDlet suite that you already have installed on the device, the emulator will replace the MIDlet suite's code and resources, but by default it will not change or remove any persistent data that the MIDlet suite had stored.

If you want to remove any persistent data written by previous versions of the MIDlet suite, use the `-removeRMS` option. The following example shows the TicketLand MIDlet suite being installed, and persistent data from previous versions being removed:

```
c:\midp2.0fcs> bin\midp -install -domain trusted
-removeRMS http://localhost:8080/j2me/midp/auction.jad
Storage name:
#Sun%0020#Microsystems%002c%0020#Inc%002e_#Ticket%0020#Auction_
```

Installing a MIDlet suite does not display the device skin or run the MIDlet after installing it. Installing a MIDlet suite downloads the JAR and JAD files and makes the MIDlet suite available to be run. (See [Chapter 1, “Using the Graphical User Interface”](#) for more information.)

Listing Installed MIDlet Suites

In order to remove or run a particular installed MIDlet suite, you need to know its number or storage name. Each MIDlet suite is assigned a number and a storage area when it is installed. The storage name of a MIDlet suite is the name of the suite's storage area. To get this information about a MIDlet using the `midp` command, use the `-list` or `-storageNames` option.

For example, the following command lists the installed MIDlet suites by number:

```
c:\midp2.0fcs> bin\midp -list
[1]
  Name: SunSamples - Games
  Vendor: Sun Microsystems, Inc.
  Version: 2.0
  Description: Sample suite of games for the MIDP.
  Storage name:
    #Sun%0020#Microsystems%002c%0020#Inc%002e_#Sun#Samples%0020%002d
%0020#Games_
  Size: 27K
  Installed From: http://localhost:8080/midlets/games.jad
  MIDlets:
    TilePuzzle
    WormGame
[2]
  Name: Ticket Auction
  Vendor: Sun Microsystems, Inc.
  Version: 2.0
  Description: Ticket Auction demo.
  Storage name:
    #Sun%0020#Microsystems%002c%0020#Inc%002e_#Ticket%0020#Auction_
  Size: 39K
  Installed From: http://localhost:8080/midlets/auction.jad
  MIDlets:
    TicketLand
```

The following example shows a command listing the installed MIDlet suites by storage name:

```
c:\midp2.0fcs> bin\midp -storageNames
#Sun%0020#Microsystems%002c%0020#Inc%002e_#Sun#Samples%0020%002d%0020#
Games_
#Sun%0020#Microsystems%002c%0020#Inc%002e_#Ticket%0020#Auction_
```

Listing MIDlet suites does not display the device skin.

Removing an Installed MIDlet Suite

When you uninstall a MIDlet suite, you delete any data that it had stored with the device emulator. When you next run the emulator, the removed MIDlet suite and its MIDlets will no longer appear on the application selector screen.

You uninstall a MIDlet suite by number or storage name. See [“Listing Installed MIDlet Suites” on page 30](#) for information on how to get these pieces of information. For example, if the example Demo MIDlet suite had a suite number of two, you would enter the following command to remove it:

```
c:\midp2.0fcs> bin\midp -remove 2
```

As another example, if the example Games MIDlet suite had a storage name of `#Sun%0020#Microsystems%002c%0020#Inc%002e_#Sun#Samples%0020%002d%0020#Games_`, you would enter the following command to remove it:

```
c:\midp2.0fcs> bin\midp -remove
                #Sun%0020#Microsystems%002c%0020#Inc%002e_#Sun#Samples%0020%002d%0020#Games_
```

Removing a MIDlet suite does not display the device skin.

Using MIDP Security Features

The *MIDP 2.0 specification* defines a security model that requires MIDlets to have permission to use security-sensitive APIs. This is different from the MIDP 1.0 security model, which had all MIDlet suites operate in a sandbox that prevented access to sensitive APIs or functions of the device.

This chapter explains the security model as the MIDP Reference Implementation implements it, and how to use it. It has the sections:

- [Overview](#)
- [The Authorization Process](#)
- [Managing the Security Policy](#)

Overview

This section defines the terms used in the security model. It has the topics:

- [Trusted and Untrusted MIDlets](#)
- [Permissions](#)
- [Protection Domains](#)

Trusted and Untrusted MIDlets

The security model defines two types of MIDlet suites:

- *Untrusted* — MIDlet suite for which the origin and the integrity of the JAR file cannot be trusted by the device (for example, MIDlet suites written for MIDP 1.0 or other unsigned MIDlet suites)
- *Trusted* — MIDlet suite with a JAR file that is both signed with a certificate that the device can verify and has not been tampered with, or unsigned MIDlet suite for which a user has explicitly set the domain with the `-domain` option to the `midp` command

An untrusted MIDlet suite runs in a restricted environment where access to protected APIs or functionality is either not allowed or is allowed only with explicit user permission. A trusted MIDlet suite can be run in a less restricted environment.

The MIDP Reference Implementation determines whether it can trust a signed MIDlet suite by following the *certificate chain* of suite's digital signature. A certificate chain is a series of certificates: the first certificate contains the public key that can be used to check the signature of the JAR file; the second certificate, if present, vouches that the first certificate is valid, and the third certificate, if present, vouches that the second certificate is valid, and so on until a *root certificate* is reached. The root certificate does not have another entity vouching for it; it is *self-signed*. (It is also possible that the first certificate is self-signed. That is the simplest certificate chain.) The certificate chain, except for the root certificate, is in the MIDlet suite's JAD file. The entity at the root of the certificate chain must have its information present on the device. The following figure shows an example certificate chain:

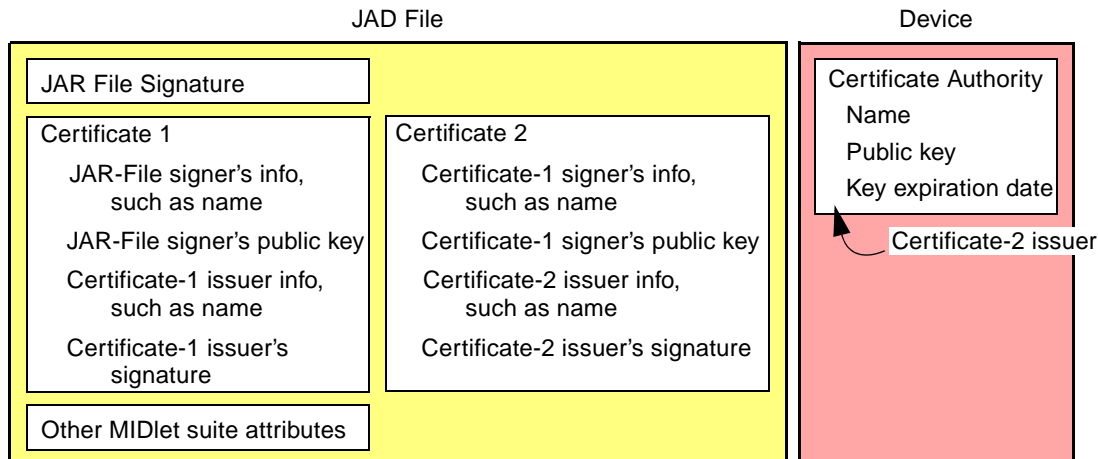


FIGURE 31 Example Certificate Chain

Root certificates are issued and signed by *certificate authorities* (CAs), which are well known, trusted entities. MIDlet developers might also use root certificates from network operators. For testing purposes, a developer can use a less official certificate as a root certificate, such as *dummyca* from the JCA keystore at *midpInstallDir/bin/j2se_test_keystore.bin* (the *readme.txt* file in that directory has the passwords) or a certificate generated using the *keytool* utility of the Java™ 2 Platform, Standard Edition (J2SE™). See <http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html> for more information.)

By being the root of a certificate chain, the entity says that it knows that the MIDlet is a legitimate application, not a malicious one. The root signer of a certificate chain must be one accepted by the device (in this case, the MIDP Reference

Implementation's device emulator) or the MIDlet suite will not be installed or run. (See [Chapter 4, "Managing Public Keys of Certificate Authorities"](#) for more information on adding a CA's public key to the device emulator.)

A trusted MIDlet suite can be granted access to security-sensitive APIs and functionality, within the bounds of the device emulator's *security policy*. A security policy associates MIDlets with *protection domains*. Protection domains list the security-sensitive APIs or functionalities that the device will allow the associated MIDlet suites to access, sometimes only after getting approval from the user. They are lists of *permissions*. (See ["Protection Domains" on page 36](#) for more information.)

Permissions

Permissions identify the security-sensitive APIs or functionalities to which the device limits access. The name of a permission indicates what it is protecting. If a permission is protecting functionality provided by a package, the permission has the package name as a prefix. If a permission is protecting functionality provided by a class, its name must include both the package and class name. For example, a permission that protects the functionality provided by the `PushRegistry` class (the class that a MIDlet uses to have the device launch it to receive an incoming message; in other words, to use the *push* functionality) is called `javax.microedition.io.PushRegistry`.

The *MIDP 2.0 specification* defines the following permissions:

TABLE 1 Permission Names Defined in the *MIDP 2.0 Specification*

Functionality	Permission Name
HTTP protocol	<code>javax.microedition.io.Connector.http</code>
HTTPS protocol	<code>javax.microedition.io.Connector.https</code>
Datagram protocol	<code>javax.microedition.io.Connector.datagram</code>
Datagram server protocol (without host)	<code>javax.microedition.io.Connector.datagramreceiver</code>
Socket protocol	<code>javax.microedition.io.Connector.socket</code>
Server socket protocol (without host)	<code>javax.microedition.io.Connector.serversocket</code>
SSL protocol	<code>javax.microedition.io.Connector.ssl</code>
Comm protocol	<code>javax.microedition.io.Connector.comm</code>
Use the registry that holds information on MIDlets that will be launched to handle incoming messages	<code>javax.microedition.io.PushRegistry</code>

Protection Domains

A protection domain defines a set of permissions that can be granted to MIDlet suites. The set does not contain every permission available, but lists only the permissions for which the protection domain can grant access. If a permission does not appear in a protection domain, then the MIDlet suites associated with the domain are denied access to that protected functionality.

To explain protection domains more fully, this section has the topics:

- [Interaction Modes](#)
- [Built-in Protection Domains](#)
- [Associating MIDlet Suites and Protection Domains](#)

Interaction Modes

A protection domain defines the maximum level of access a MIDlet suite can be granted to each permission in its permission set. Each can be either an:

- *Allowed permission* — MIDlet suites in the protection domain can use the API or functionality without involving the user.

The name for this interaction mode is `Allowed`.

- *User permission* — MIDlet suites in the protection domain can use the API or functionality only after the user explicitly grants permission. If the user denies permission, the MIDlet cannot run the protected code. Users can grant permission with one of the following interaction modes:
 - *Blanket* — Permission is granted for every invocation of the API by a MIDlet suite until it is uninstalled or the user changes the permission.
 - *Session* — Permission is granted until the user exits the MIDlet suite. If the user restarts the MIDlet suite, the permission request is repeated.
 - *Oneshot* — Permission is granted for only that one call. Each time the MIDlet calls the protected code, the permission request is repeated.

The user permission interaction modes in the protection domain are the highest levels of access that a user may grant to a MIDlet suite. A protection domain can also associate a default interaction mode with a user permission. The default is the interaction mode that the device could suggest to a user. For example, consider a device that requests permission by presenting interaction modes in an exclusive choice list. (The interaction modes might be presented in choices like, “No, Not this time,” “Yes this time, and ask me again next time,” “Yes from now on. Don’t ask again,” and so on.) In this case, the device would use the default to determine which element of the list to preselect.

Built-in Protection Domains

The MIDP Reference Implementation provides the following protection domains:

- *Minimum* — Denies its MIDlet suites access to all security-sensitive APIs; this domain contains no permissions
- *Untrusted* — Requires that its MIDlet suites get user permission to use the push functionality and the network protocols HTTP, HTTPS, socket, datagram, server sockets, and comm
- *Trusted* — Allows access to the push functionality and all network protocols implemented by this product
- *Maximum* — Same as trusted

As the domain descriptions illustrate, the lists of permissions associated with a protection domain can be long. To make protection domains easier to manage, they can contain *aliases*. An alias is a name for a group of permissions. When a protection domain uses an alias for one or more of its permissions, it grants the same level of access to all the permissions in that alias.

Associating MIDlet Suites and Protection Domains

The MIDP Reference Implementation treats signed and unsigned MIDlet suites differently when assigning protection domains. If a MIDlet suite is unsigned, you can force it into any domain when you use the command-line to install it or temporarily download and run it. (Using the command-line is described in [Chapter 2, “Using the midp Executable.”](#)) If it is not forced, the MIDP Reference Implementation assigns an unsigned MIDlet suite to the untrusted domain.

If the MIDlet suite is signed, the MIDP Reference Implementation will assign it to the domain associated with an entity that signed the suite. That is, the device emulator holds the public keys of certificate authorities. When you enter a public key into the device emulator, you associate the key with a domain. (See [“Importing a Key” on page 44](#) for more information.) When the MIDP Reference Implementation checks the MIDlet suite’s digital signature, it uses a CA’s public key. It then assigns the MIDlet suite to the protection domain associated with that public key, if it determines that it can trust the suite.

The Authorization Process

Authorization is an interaction between the needs of the MIDlet suite to use protected APIs, and the permission policies of the device. For trusted MIDlet suites there are two times that the MIDP implementation checks permissions. The first is when the user tries to put the MIDlet suite on the device; the second is when a MIDlet in the suite calls a protected API:

- Putting the MIDlet on the device – The MIDP implementation checks the set of permissions requested by the MIDlet suite against the permissions that the suite’s protection domain could grant. (See the previous section, “[Associating MIDlet Suites and Protection Domains](#)” on page 37 for how the device matches a MIDlet suite with a protection domain.) If the MIDlet suite requires a permission that the device cannot grant, the device does not accept the MIDlet suite.
- When a MIDlet from a suite tries to call a protected API or use protected functionality, the MIDP implementation checks whether the MIDlet has been granted access to it. If it must, it queries the user about whether to grant the permission. If the MIDlet is granted access (a successful authorization), the MIDP implementation runs the protected code. If the MIDlet has not been granted access, the MIDP implementation throws a `java.lang.SecurityException`.

MIDP checks the permissions of untrusted MIDlets only when they call a protected API.

Managing the Security Policy

The MIDP Reference Implementation defines its security policy in the file `midpInstallDir\appdb_policy.txt`, where `midpInstallDir` is the directory that holds your MIDP installation. The policy file defines the following aliases, to make the file easier to read and maintain:

- `net_access` – Permissions for protocols that enable the device emulator to access network resources
- `application_auto_invocation` – Permission to register with the device so that it can be started in response to an incoming message
- `local_connectivity` – Permission to use the comm protocol

It also has the four domains listed previously, minimum, maximum, trusted, and untrusted. In the untrusted domain, the permissions have a user permission of blanket and a default level of session. [CODE EXAMPLE 1](#) shows the policy file.

CODE EXAMPLE 1 The *midpInstallDir*\appdb_policy.txt File

```
alias: net_access
    javax.microedition.io.Connector.http,
    javax.microedition.io.Connector.socket,
    javax.microedition.io.Connector.https,
    javax.microedition.io.Connector.ssl,
    javax.microedition.io.Connector.datagram,
    javax.microedition.io.Connector.serversocket,
    javax.microedition.io.Connector.datagramreceiver

alias: application_auto_invocation
    javax.microedition.io.PushRegistry

alias: local_connectivity
    javax.microedition.io.Connector.comm

domain: minimum

domain: maximum
    allow: net_access
    allow: application_auto_invocation
    allow: local_connectivity

domain: trusted
    allow: net_access
    allow: application_auto_invocation
    allow: local_connectivity

domain: untrusted
    blanket(session): net_access
    blanket(session): application_auto_invocation
    blanket(session): local_connectivity
```

You can change this file with any text editor.

Note – Do not change this file in a way that is incompatible with the *MIDP 2.0 Specification* or with the MIDP TCK.

The *MIDP 2.0 Specification* mandates that a MIDP 2.0 implementation be able to run MIDP 1.0 MIDlets and keep the HTTP and HTTPS functionality protected. This means that the untrusted domain must have a user level of access for the following permissions:

```
javax.microedition.io.Connector.http
javax.microedition.io.Connector.https
```

See [TABLE 1 on page 35](#) for the list of permissions defined by the *MIDP 2.0 Specification*. See the documentation that comes with your TCK for its security requirements.

Managing Public Keys of Certificate Authorities

This chapter describes how to manage the certificate authority (CA) keys that your MIDP implementation uses to ensure that you access secure, valid web sites and to ensure that it properly accepts and classifies signed MIDlet suites.

This chapter contains the sections:

- [Overview](#)
- [General Instructions for MEKeyTool](#)
- [Working With Multiple ME Keystores](#)
- [Importing a Key](#)
- [Listing Available Keys](#)
- [Deleting a Key](#)
- [Replacing a Key](#)
- [Handling Certificate Exceptions When Running MIDP](#)

See [Appendix B, “The MEKeyTool Utility”](#) for information in a manpage format on managing certificate authority keys.

Overview

The MIDP Reference Implementation uses the public keys of CAs for two reasons. One is to check the validity of web sites, and the other is to check the validity of a signed MIDlet suite. (See [Chapter 3, “Using MIDP Security Features”](#) for more information on checking the validity of signed MIDlet suites.)

When you use a secure protocol to access a web site, the site provides a certificate. The certificate is typically signed by a CA. By signing the certificate, the CA is saying that the web site is being run by the owners of the certificate. (It is not, for example, a look-alike web site set up by hackers to collect information.)

MIDP Reference Implementation checks the web site’s certificate using the public key of the CA that signed it. If the certificate is valid, you are permitted to visit the site. If there is a problem with the web site’s certificate, or you do not have the

public key of the CA that signed the certificate, you will not be permitted to access the site. For example, if you try to connect to a test site with a self-signed certificate, and you do not have access to that site's public key, you will receive a certificate error.

To manage public keys for the MIDP Reference Implementation, you use the `MEKeyTool` utility. The utility is functionally similar to the `keytool` utility that comes with the Java™ 2 Platform, Standard Edition (J2SE™). The `MEKeyTool` utility enables you to:

- See which CA public keys are available to MIDP Reference Implementation.
- Add new CA public keys to those available to MIDP Reference Implementation.
- Replace expired CA public keys that MIDP Reference Implementation was trying to use.
- Remove CA public keys that you no longer want MIDP Reference Implementation to use.

The `MEKeyTool` utility keeps the CA public keys in an *ME keystore*, a file that holds the keys in a format that Java 2 Platform, Micro Edition (J2ME™) can use. By contrast, a file that holds the keys in a format that the J2SE platform can use is a *JCA keystore*.

Note that JCA and ME keystores have different formats. You cannot use the `MEKeyTool` utility on a JCA keystore. If you do (for example, to try to see its public keys), you will receive an error message that the keystore is corrupted. The message means that the JCA keystore is not in a format that the `MEKeyTool` utility can read. The JCA keystore is in the correct format for the J2SE platform tools.

General Instructions for MEKeyTool

The `MEKeyTool` utility is packaged in a JAR file, `MEKeyTool.jar`, in the `midpInstallDir\bin` directory. To use the utility:

- 1. Open a command prompt or terminal window.**

- 2. Change your current directory to `midpInstallDir`.**

For example, if MIDP Reference Implementation was installed in the directory

```
c:\midp2.0fcs:
```

```
c:\> cd midp2.0fcs
```

- 3. Run `MEKeyTool` with the `java` command, the `-jar` option, and any commands or options to `MEKeyTool`.**

The commands and options to the `MEKeyTool` utility are described below. Using no options provides help on `MEKeyTool`. For example, if the `java` command were on your PATH, you could get help with the command:

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar
```

Working With Multiple ME Keystores

When you use a secure protocol, the MIDP Reference Implementation expects its ME keystore to be the file *midpInstallDir\appdb_main.ks*. The file is included with MIDP Reference Implementation, and contains the key of one popular CA. By default the MEKeyTool utility uses this file when you add, delete, or list keys.

You can also use the MEKeyTool utility to manage other ME keystores. For example, during testing you might want to have multiple keystores to run against: one might have all the keys that you need, another might be missing one or more keys, another might have an expired key. You would use the MEKeyTool utility to manage all of them.

This section has the following topics:

- [Creating Alternate ME Keystores](#)
- [Managing Alternate ME Keystores](#)
- [Running MIDP and Alternate Keystores](#)

Creating Alternate ME Keystores

Importing a key is a one way to create a new ME keystore: if you import a key into a keystore that does not exist, the MEKeyTool utility creates it. See [“Importing a Key” on page 44](#) for instructions on how to import a key. Another way to create an ME keystore is to copy the keystore provided with MIDP Reference Implementation.

Managing Alternate ME Keystores

To manage a keystore other than the default, use the -MEkeystore option to MEKeyTool. That is, the command will begin like this:

```
java -jar bin/MEKeyTool.jar -MEkeystore keystoreName ...
```

For example, assume you have created an ME keystore that contains the keys that you need for a particular set of tests, *c:\myKeys\set2_test_keys.ks*. The command to run the MEKeyTool utility to manage that keystore would begin like this:

```
java -jar bin/MEKeyTool.jar  
-MEkeystore c:/myKeys/set2_test_keys.ks ...
```

For most MEKeyTool commands, if the file that you provide as an argument to -MEkeystore does not exist you will receive an error message. The one exception is importing a key. In this case MEKeyTool creates a new ME keystore with the name that you supply.

Running MIDP and Alternate Keystores

Note – The MIDP Reference Implementation executable uses only the ME keystore `midpInstallDir\appdb_main.ks`.

Because the MIDP executable uses only the ME keystore `midpInstallDir\appdb_main.ks`, you need to do the following steps *before* running MIDP if you want to use an alternate keystore:

1. **Back up the existing `midpInstallDir\appdb_main.ks`, if necessary.**

If you already have a copy of this file, go to the next step.

For example, assuming that `midpInstallDir` is `c:\midp2.0fcs`, you would run a command such as this:

```
c:\midp2.0fcs> cp appdb/_main.ks c:/myKeys/orig_main.ks
```

2. **Copy your alternate keystore to `midpInstallDir\appdb_main.ks`.**

For example, if you want to use the keys in the file `c:\myKeys\set2_test_keys.ks` for this run of the MIDP Reference Implementation executable you would execute a command like the following one:

```
c:\midp2.0fcs> cp c:/myKeys/set2_test_keys.ks appdb/_main.ks
```

Importing a Key

Because the ME keystore comes with few keys, you might find that you cannot visit some web sites or install some MIDlets until you add more keys to the keystore. You will also have to add keys to the keystore as time passes and keys expire. (You would first delete the expired key, then add the new one.)

You add a key to an ME keystore by importing it from a JCA keystore. You can import keys from the JCA keystore that comes with the J2SE platform or from a JCA keystore that you create. For more information on the keystore that comes with the J2SE platform, see:

<http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html>

The default JCA keystore for the `MEKeyTool` utility is `userHome\.keystore`. Whatever keystore you use, if it requires a passwords, you must provide it.

When you add a key to an ME keystore, you can also associate a security domain with it. The MIDP Reference Implementation can assign that domain to any MIDlet suite for which the owner of the public key was a signer. (See [“Protection Domains” on page 36](#) for more information.) If you don’t provide a domain, the public key is associated with the untrusted domain.

The option that imports a key is `-import`. For example, assume that you want to add a key with an alias `dummyca` from the JCA keystore `../bin/j2se_test_keystore.bin` to the ME keystore `c:\myKeys\set2_test_keys.ks`. Further assume that the JCA keystore has a password, `keystorepwd`, and that you want to assign the public key to the trusted domain. You would do this with the command:

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar -import
               -alias dummyca
               -keystore ../bin/j2se_test_keystore.bin -storepass keystorepwd
               -MEkeystore c:/myKeys/set2_test_keys.ks -domain trusted
```

Listing Available Keys

An ME keystore organizes the keys that it contains by giving each one a number. The keystore also holds, for each key, the name of the entity to whom the public key belongs, the time over which the key is valid, and the domain associated with the key. The `-list` command to the `MEKeyTool` utility shows you all this information for each key in a particular keystore.

The following example lists the contents of the ME keystore

```
c:\myKeys\set1_test_keys.ks:
```

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar -list
               -MEkeystore c:/myKeys/set1_test_keys.ks
```

Key 1

```
Owner: C=US;O=RSA Data Security, Inc.;OU=Secure Server
      Certification Authority
```

```
Valid from Tue Nov 08 16:00:00 PST 1994 to Thu Jan 07 15:59:59 PST
      2010
```

```
Security Domain: untrusted
```

Key 2

```
Owner: O=Sun Microsystems;C=myserver
```

```
Valid from Sat Aug 03 00:43:51 PDT 2002 to Tue Jul 31 00:43:51 PDT
      2012
```

```
Security Domain: trusted
```

Deleting a Key

As keys expire, you will have to delete them from the keystore so that you can add their replacements. You might also want to delete keys that you will no longer be using. (For example, if you add the public key of a test site with a self-signed certificate, you might want to remove that key when testing is over.)

The `-delete` command to the `MEKeyTool` utility removes a key from an ME keystore. The command requires one of the following options:

- `-owner ownerName`

String that describes the owner of the public key. The string must match the one that is printed when you use the `-list` command to the `MEKeyTool` utility. (See the previous section, [“Listing Available Keys”](#) for more information.)

- `-number keyNumber`

Number greater than or equal to one. The `-list` command prints the number that a given keystore has assigned to each of its keys. (See the previous section, [“Listing Available Keys”](#) for more information.)

The following examples show the two ways to delete a key from the ME keystore `c:\myKeys\set1_test_keys.ks` used in the previous section, [“Listing Available Keys.”](#) The first example deletes the key using its key number:

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar -delete -number 1
-MEkeystore c:/myKeys/set1_test_keys.ks
```

The second example deletes the key using its owner name:

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar -delete -owner "OU=Secure
Server Certification Authority;O=RSA Data Security, Inc.;C=US" -
MEkeystore c:/myKeys/set1_test_keys.ks
```

Replacing a Key

Replacing keys is necessary in some situations, such as when a key expires. (See [“Handling Certificate Exceptions When Running MIDP,”](#) next, for other situations where key replacement is in order.)

To replace a key you must first delete the old key, then import the new key. (If you try to import the new key before deleting the old one, you will receive an error message telling you that the owner of the key already has a key in the ME keystore.)

See [“Deleting a Key” on page 46](#) for instructions on how to delete a key, and [“Importing a Key” on page 44](#) for instructions on how to import the new one.

Handling Certificate Exceptions When Running MIDP

When you run MIDP, you could encounter errors with certificates. When methods that authenticate a certificate encounter such a problem, they throw a `CertificateException`. Each `CertificateException` contains a reason code that indicates the nature of the problem. If you receive a `CertificateException`, you can get its reason code by calling its `getReason` method.

The following list contains the reason codes followed by their corrective actions and descriptions.

- `BAD_EXTENSIONS` — Replace the end entity's certificate. The certificate has a field that specifies additional critical information that the system does not recognize. (The system must reject certificates that contain unrecognized critical extensions.)
- `BROKEN_CHAIN` — Replace the end entity's certificate chain. The chain has a certificate that was not issued by the next authority in the chain.
- `CERTIFICATE_CHAIN_TOO_LONG` — Replace the end entity's certificate chain with a shorter one. (The CA specifies the maximum length of the end entity's certificate chain.)
- `EXPIRED` — Replace the certificate because the current date is past the last date on which the certificate is valid.
- `INAPPROPRIATE_KEY_USAGE` — Replace the end entity's certificate. The certificate has an invalid value in a field that indicates the purposes for which the certificate can be used. (The field names are `keyUsage` or `extendedKeyUsage`. They are typically critical extensions.)
- `MISSING_SIGNATURE` — Replace the end entity's certificate with one that has a signature; the current certificate is unsigned.
- `NOT_YET_VALID` — Replace the certificate because the current date is before the date that the certificate can be used.
- `ROOT_CA_EXPIRED` — Replace the CA's public key on the device using the `MEKeyTool` utility. The current date is past the last date on which the CA's public key is valid.
- `SITENAME_MISMATCH` — Replace the end entity's certificate with one that has the same site name as the common name attribute of subject name.
- `UNAUTHORIZED_INTERMEDIATE_CA` — Replace certificate chain because an intermediate certificate in the chain does not have the authority to be a intermediate CA.
- `UNRECOGNIZED_ISSUER` — Import the public key of the end entity's CA into the device using the `MEKeyTool` utility. The system can't recognize the end entity without that key.

- `UNSUPPORTED_PUBLIC_KEY_TYPE` — Replace the end entity's certificate because its public key is not supported by the device.
- `UNSUPPORTED_SIGALG` — Replace the end entity's certificate with one that was signed using RSA. The signature of the current certificate uses another algorithm that this system does not support.
- `VERIFICATION_FAILED` — Replace the certificate because the system cannot confirm that the certificate is valid.

The exception's detail message, returned by the `getMessage` method, provides additional information.

The JadTool Utility

JadTool — add certificates to and see certificates in a Java™ application descriptor (JAD) file; create a digital signature of a JAR file, and add that signature to a JAD file.

Synopsis

```
java -jar JadTool.jar
[ -addcert -alias keyAlias [ -keystore keystore ] [ -storepass password ]
  [ -certnum certNumber ] [ -chainnum chainNumber ]
  [ -encoding encoding ] -inputjad inputJadFile -outputjad outputJadFile ]
[ -addjarsig [ -jarfile jarFile ] -alias keyAlias [ -keystore keystore ]
  -storepass password -keypass keyPassword
  [ -encoding encoding ] -inputjad inputJadFile -outputjad outputJadFile ]
[ -help ]
[ -showcert
  [ ( [ -certnum certNumber ] [ -chainnum chainNumber ] ) | -all ]
  [ -encoding encoding ] -inputjad inputJadFile ]
```

Description

The JadTool utility signs a JAR file, which means that it adds a certificate and the JAR file's digital signature to a JAD file. Signing a JAR file is one way that a MIDP environment that includes security might trust a MIDlet suite. Trusted MIDlet suites typically have more permissions to use protected, security-sensitive APIs and functionality than untrusted MIDlet suites.

Adding a certificate and a JAR file's digital signature to a JAD file are separate steps, both of which must be completed to sign a JAR file. A JAD file can have more than one certificate, but can hold the signature for only one JAR file. When a certificate in the JAD file expires, a new certificate must be added to the it, and the JAR file re-signed. (The JadTool utility will overwrite the digital signature it currently holds with the new signature.)

Note – It is up to the user of the `JadTool` utility to ensure that the certificate in the JAD file is from the same JCA keystore entry as the one used to sign the JAR file.

The `JadTool` utility can also provide information about a certificate in a JAD file, including the name of the entity that issued the certificate, the certificate's serial number, the dates between which it is valid, and its MD5 and SHA fingerprints.

Options

The following options are supported:

none

Running the tool without options returns the same information as the `-help` option.

```
-addcert -alias keyAlias [ -keystore keystore ] [ -storepass password ]  
[ -chainnum chainNumber ] [ -certnum certNumber ]  
[ -encoding encoding ] -inputjad inputJadFile -outputjad outputJadFile ]
```

Adds a certificate to a JAD file. To do this, this utility first creates the certificate from the entry identified by *keyAlias* in *keystore*. The *keystore*, if provided, must be a JCA keystore (a file containing data such as key entries in a format that the Java 2 Platform, Standard Edition (J2SE™) can use). If *keystore* is not provided, its default, `$HOME/.keystore`, is used. If *keystore* requires a password to access its contents, *password* must be provided.

After creating the certificate and attribute name, this utility concatenates the contents of *inputJadFile* with the new certificate and writes it as *outputJadFile*. The certificate is in the JAD file as the value of an attribute named `MIDlet-Certificate-m-n`, where:

- *m* is *chainNumber*, or 1 if it is not provided. (A JAD file can contain more than one certificate chain.)
- *n* is *certNumber*. The value *certNumber* depends on whether the new certificate will replace an existing certificate. If the certificate is a replacement, then *certNumber* should be the number of the certificate to replace. For example, if the new certificate would replace the one stored as the value of attribute `MIDlet-Certificate-1-3`, then *certNumber* should be 3. If the certificate is new, *certNumber* is ignored.

If *inputJadFile* uses an encoding other than UTF-8 (ascii with unicode escapes), *encoding* must be specified. This utility uses the same encoding for reading *inputJadFile* and writing *outputJadFile*.

```
-addjarsig [ -jarfile jarFile ] -alias keyAlias [ -keystore keystore ]  
-storepass storePassword -keypass keyPassword  
[ -encoding encoding ] -inputjad inputJadFile -outputjad outputJadFile
```

Creates a digital signature for *jarFile*. If *jarFile* is not specified, the value of the MIDlet-Jar-URL attribute from *inputJadFile* is used. (The attribute's value must be a valid HTTP URL.)

This utility creates a digital signature for the JAR file using the private key identified by *keyAlias* in *keystore*. If *keystore* is not provided, its default is \$HOME/.keystore. This utility gets the key from *keystore* using *storePassword* and *keyPassword*, and creates the signature with it using the EMSA-PKCS1-v1_5 encoding method of PKCS #1, version 2.0. (See RFC 2437 at <http://www.ietf.org/rfc/rfc2437.txt>.)

After creating the signature, this utility concatenates the contents of *inputJadFile* with the signature, and writes it as *outputJadFile*. The signature is base64 encoded, and is in the output JAD file as the value of the MIDlet-Jar-RSA-SHA1 attribute.

If *inputJadFile* uses an encoding other than UTF-8 (ascii with unicode escapes), *encoding* must be specified. This utility uses the same encoding for reading *inputJadFile* and writing *outputJadFile*.

```
-help
```

Prints a usage summary.

```
-showcert [ ( [ -certnum certNumber ] [ -chainnum chainNumber ] ) | -all ]  
[ -encoding encoding ] -inputjad inputJadFile
```

Prints information about either all certificates, or the certificate that corresponds to the given *certNumber* and *chainNumber* in the *inputJadFile*. (The option *-all* cannot be combined with the *-certnum* and *-chainnum* options.) The *chainNumber* of a certificate is the *m* in the JAD file's MIDlet-Certificate-*m-n* attribute, while the *certNumber* is the *n*. For example, to show the certificate that is the value of attribute MIDlet-Certificate-2-3, then *chainNumber* should be 2 and *certNumber* should be 3. If *certNumber* or *chainNumber* are not provided, this utility uses a 1.

The information printed includes the certificate's subject, issuer, serial number, dates between which it is valid, and fingerprints (md5 and SHA). The attributes in the subject and issuer names are shown in reverse order from what is in the certificate (a side effect of using the J2SE platform certificate API). As a result, the names may not match what is returned from other tools that display a certificate's subject and issuer names.

If *inputJadFile* uses an encoding other than UTF-8 (ascii with unicode escapes), *encoding* must be specified. The tool uses the same encoding for reading *inputJadFile* and writing *outputJadFile*.

Examples

In the following examples the JCA keystore is `midpInstallDir\bin\j2se_test_keystore.bin`. The password for the keystore is `keystorepwd`. The keystore has RSA key pair with an alias of `dummyca`; it has a password of `keypwd`.

Adding a Certificate to a JAD file

The following command creates a new JAD file that contains the contents of the input JAD file and a certificate created from the specified key:

```
c:\midp2.0fcs> java -jar bin/JadTool.jar -addcert
               -alias dummyca -keystore bin/j2se_test_keystore.bin
               -storepass keystorepwd -inputjad example/pushdemo.jad
               -outputjad example/pushdemo-w-mykey.jad
```

Viewing a Certificate

The following command shows the certificate associated with certificate number two (and, by default, chain number one) in the named JAD file:

```
c:\midp2.0fcs> java -jar bin/JadTool.jar -showcert
               -certnum 2 -inputjad example/pushdemo-w-mykey.jad
Subject: C=myserver, O=Sun Microsystems
Issuer : C=myserver, O=Sun Microsystems
Serial number: 3cf7e309
Valid from Sun Jul 28 15:06:50 PDT 2002 to Wed Jul 25 15:06:50 PDT
          2012
Certificate fingerprints:
    MD5: df:b1:40:a7:3c:4d:5b:f5:91:27:68:86:9c:a6:68:9e
    SHA: 20:c4:1e:53:cc:cd:d0:7e:eb:a3:64:57:c2:1f:eb:88:23:cc:aa:25
```

Signing a JAR File

The following command signs a JAR file:

```
c:\midp2.0fcs> java -jar bin/JadTool.jar -addjarsig
               -alias dummyca -keystore bin/j2se_test_keystore.bin
               -storepass keystorepwd -keypass keypwd
               -jarfile example/pushdemo.jar
               -inputjad example/pushdemo-w-mykey.jad
               -outputjad example/pushdemo-w-certsig.jad
```

See Also

Creating MIDlet Suites

The MEKeyTool Utility

MEKeyTool — manage a file (an *ME keystore*) that contains the public keys of trusted certificate authorities (CAs).

Synopsis

```
java -jar MEKeyTool.jar
  -delete [ -MEkeystore MEKeystore ]
    ( -owner ownerName | -number keyNumber )
  [-help]
  -import [ -MEkeystore MEKeystore ] -alias keyAlias
    [ -keystore JCAKeystore ] [ -storepass storePassword ]
    [ -domain domain ]
  -list [ -MEkeystore MEKeystore ]
```

Description

The `meKeyTool` utility manages one or more ME keystores, which are files that hold the public keys of trusted certificate authorities (CAs). It can be used to create a new ME keystore, or to add to, delete from, or list the contents of an existing keystore.

The `MEKeyTool` gets the public keys to add to an ME keystore from a Java™ Cryptography Architecture (JCA) keystore. Such a keystore is part of the Java 2 Platform, Standard Edition (J2SE™); see the J2SE platform documentation for more information.

Note that the ME keystore and the JCA keystore have different formats. The `MEKeyTool` cannot be used to manage a JCA keystore. If the `MEKeyTool` utility is used with a JCA keystore (for example, to try to list its keys), you will receive an error message that the keystore is corrupted. The JCA keystore is not corrupted; it can still be managed with J2SE platform tools. The keystore is just not in a format the `MEKeyTool` can read.

If it is necessary to replace a key in an ME keystore, the key must be removed and the new key entered. There is not a single command to do a replacement, and the tool does not permit adding a different version of an existing key.

Although the `MEKeyTool` manages ME keystores of any name, the MIDP Reference Implementation executable expects its ME keystore to be the file `midpInstallDir\appdb_main.ks`, where `midpInstallDir` is the location of the MIDP installation. If there are multiple keystores, make sure the one to be used by MIDP has the name `_main.ks` in the `midpInstallDir\appdb` directory before running the MIDP executable.

Options

The following options are supported:

none

Running the tool without options returns the same information as the `-help` option.

`-delete [-MEkeystore MEKeystore]
(-owner ownerName | -number keyNumber)`

Deletes the key with *ownerName* or *keyNumber* from *MEKeystore*. If *MEKeystore* is not provided, its default, `appdb/_main.ks`, is used.

You can provide either *ownerName* or *keyNumber*, but not both. You can find the valid values for them by running the `MEKeyTool` utility with the `-list` command.

`-help`

Prints a usage summary.

`-import [-MEkeystore MEKeystore] -alias keyAlias
[-keystore JCAKeystore] [-storepass storePassword] [-domain domain]`

Imports a public key from *JCAKeystore* into *MEKeystore*, and associates the public key with *domain*. If *JCAKeystore* is not provided, its default, `userHome\.keystore`, is used (where *userHome* is the user's home directory). If *MEKeystore* is not provided, its default, `appdb/_main.ks`, is used. If *domain* is not provided, its default, `untrusted`, is used.

If *JCAKeystore* requires a password, you must provide *storePassword*.

`-list [-MEkeystore MEKeystore]`

Lists the number, owner, and validity period, and domain of each key in *MEKeystore*. If *MEKeystore* is not provided, the default, `appdb/_main.ks`, is used.

Examples

Listing Keys

To see the keys in an ME keystore, use the `-list` option. Provide the name of the ME keystore if it is not `midpInstallDir\appdir_main.ks`, where `midpInstallDir` is the location of the MIDP installation.

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar -list
-MEkeystore c:/myKeys/test_keys.ks
Key 1
  Owner: O=Sun Microsystems;C=myserver
  Valid from Sat Aug 03 00:43:51 PDT 2002 to Tue Jul 31 00:43:51 PDT
        2012
  Security Domain: trusted
```

Adding a Key

Adding a key to an ME keystore requires that the key be in a JCA keystore. Provide the password to the JCA keystore if it requires one, and the alias for the key. Also provide the name of the ME keystore if it is not `midpInstallDir\appdir_main.ks`, where `midpInstallDir` is the location of the MIDP installation, and the domain if it should be something other than `untrusted`.

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar -import -alias dummyca
-keystore bin/j2se_test_keystore.bin -storepass keystorepwd
-MEkeystore c:/myKeys/test_keys.ks -domain trusted
```

Deleting a Key

Deleting a key from an ME keystore requires a key's number in the ME keystore or its owner, both of which can be obtained by listing the contents of the ME keystore. When deleting a key, provide the name of the ME keystore if it is not `midpInstallDir\appdir_main.ks`, where `midpInstallDir` is the location of the MIDP installation. For example, to delete key number one:

```
c:\midp2.0fcs> java -jar bin/MEKeyTool.jar -delete -number 1
-MEkeystore c:/myKeys/test_keys.ks
```

See Also

[Chapter 4, “Managing Public Keys of Certificate Authorities”](#)

The midp Command

`midp` — device emulator and Java™ programming language interpreter for the Mobile Information Device Profile (MIDP).

Synopsis

```
midp [ -classpath path ] [ -Dproperty=value ] [ -heapsize size ]  
command-switch
```

```
midp_g [ -classpath path ] [ -Dproperty=value ] [ -heapsize size ]  
command-switch
```

Description

The `midp` command runs a device emulator and executes bytecodes that have been created by the compiler, `javac`, and preverified by the `preverify` utility. The `midp_g` command is a version of the `midp` command built with debug capabilities. Unless otherwise noted, running the `midp` or `midp_g` command displays a device skin.

MIDP is part of the Java 2 Platform, Micro Edition (J2ME™). MIDP was initially defined in the Mobile Information Device Profile for the J2ME Platform [JSR-000037], and refined in the Mobile Information Device Next Generation [JSR-000118]. For more information on the JSRs, see <http://www.jcp.org/jsr/tech/j2me.jsp>

Applications that the `midp` command runs are called *MIDlets*. When one or more MIDlets are packaged as a unit for download onto a device, the bundle is called a *MIDlet suite*. The `midp` command can run MIDlets and MIDlet suites.

In addition to running MIDlets and MIDlet suites, the `midp` command manages MIDlet suites. It can download and install them on the emulator, list the MIDlet suites installed on the emulator, remove a MIDlet suite from the emulator, and so on.

The behavior of the emulator during MIDlet execution is controlled by configuration files:

- *midpInstallDir*\build\share\lib\internal.config — Properties that enable MIDP to better simulate the capabilities of a device (such as number of colors).
- *midpInstallDir*\build\share\lib\system.config — Properties that describe the MIDP environment (such as the version number of the MID profile).

Where *midpInstallDir* is the directory where MIDP is installed. The files are property files that use a colon to separate the property name from its value:

propertyName : *propertyValue*

White space around the colon is not significant.

The tables below describe the properties in each file.

TABLE 2 Properties in the internal.config File

Property Name	Type	Default Value	Description
system.jam_space	integer	4194304	Amount of space to reserve for MIDlet Suites. This includes storage of the MIDlet Suites themselves as well as any RMS storage. The value is specified in bytes.
system.i18n.lang	String	en	Default language for I18N processing.
system.i18n.encoding	String	ISO8859_1	Default encoding for I18N processing.
system.display.double_buffered	boolean	true	Whether to use double buffering when writing to the graphics display.
system.display.screen_depth	integer	4	Number of bits to use for each color. The maximum number of colors is 2 to the power of this number. Set to 1, 2, 4 or 8 for 2, 4, 16 or 256 colors respectively.
system.display.visual_type	String	" "	Visual class of window. Value can be one of: <ul style="list-style-type: none"> • StaticGray • GrayScale • StaticColor • PseudoColor • TrueColor • DirectColor This property is valid only on the Solaris™ Operating Environment (OE).

TABLE 2 Properties in the `internal.config` File (Continued)

Property Name	Type	Default Value	Description
<code>com.sun.midp.midlet. platformRequestCommand</code>	String	<code>" "</code>	Command spawned when the method <code>MIDlet.platformRequest</code> is called. The command should take a single argument, a URL; it will be given the URL passed to the <code>MIDlet.platformRequest</code> method.
<code>com.sun.midp.io.http.proxy</code>	String	<code>" "</code>	Host and port of the HTTP proxy in the format of <i>host:port</i> . The value of this property is also used to tunnel HTTPS connections.
<code>com.sun.midp.io.http. force_non_persistent</code>	boolean	<code>false</code>	Force all connections to be non persistent.
<code>com.sun.midp.io.http. max_persistent_connections</code>	integer	<code>1</code>	Maximum number of persistent connections allowed at any given time.
<code>com.sun.midp.io.http. persistent_connection_linger_ time</code>	Integer	<code>60000</code>	Number of milliseconds an unused persistent connection should remain in the connection pool.
<code>com.sun.midp.io.http. inputBufferSize</code>	integer	<code>256</code>	The size, in bytes, of the input buffer. There is an input buffer for every connection.
<code>com.sun.midp.io.http. outputBufferSize</code>	integer	<code>2048</code>	The size, in bytes, of the output buffer. There is an output buffer for every connection.
<code>com.sun.midp.lcdui. eventHandler</code>	String	<code>com.sun.midp.lcdui. DefaultEventHandler</code>	The name of the class that handles events.
<code>com.sun.midp.lcdui. inputMethodHandler</code>	String	<code>com.sun.midp. lcdui.i18n. DefaultInput MethodHandler_ locale</code>	The name of the class to handle input. The default input handler uses the value of the <code>microedition.locale</code> attribute to determine the correct class name.

TABLE 2 Properties in the `internal.config` File (Continued)

Property Name	Type	Default Value	Description
<code>com.sun.midp.midlet.scheduler</code>	String	<code>com.sun.midp.midlet.Scheduler</code>	The name of the class used to schedule MIDlets.
<code>com.sun.midp.graphicalmanager</code>	String	<code>com.sun.midp.dev.Manager</code>	The name of the class used to manage MIDlet Suites in the device emulator's graphical user interface.
<code>com.sun.midp.midletsuite.installer</code>	string	<code>com.sun.midp.midletsuite.Installer</code>	Name of the class used to install MIDlet suites.

TABLE 3 Properties in the `system.config` File

Property Name	Type	Default Value	Description
<code>microedition.configuration</code>	String	CLDC-1.0	Version number of the underlying J2ME platform configuration. When there are multiple configurations, they will be separated by blanks (Unicode x20)
<code>microedition.encoding</code>	String	ISO-8859-1	Current character encoding
<code>microedition.hostname</code>	String	" "	Name of the host platform
<code>microedition.profiles</code>	String	MIDP-2.0	List of version numbers of supported J2ME platform profiles; profiles are separated by blanks (Unicode x20)
<code>microedition.locale</code>	String	" "	Current locale for I18N support.
<code>microedition.platform</code>	String	j2me	Current host platform.
<code>com.sun.midp.io.j2me.comm.bufferSize</code>	Integer	256	The size, in bytes, of the input buffer. Each comm connection has an input buffer.
<code>com.sun.midp.io.j2me.socket.bufferSize</code>	Integer	0	The size, in bytes, of the input buffer. Each socket connection has an input buffer.
<code>javax.microedition.io.Connector.protocolPath</code>	String	<code>com.sun.cldc.io</code>	Package prefix used to find protocols The protocol class name will be <code>protocolPath.j2me.protocol.Protocol.class</code> .

To permanently change a property value, edit the appropriate file. To temporarily change a property value, use the `-D` command-line option to the `midp` command.

Options

The following options are supported:

none

Running the tool without options launches the device emulator's graphical user interface.

`-classpath path`

Specifies the directories, zip files, and JAR files to search for Java class files. This argument is passed to CLDC.

`-Dproperty=value`

Overrides the value of *property* set in the configuration file. This argument is passed to CLDC.

`-heapsize size`

Specifies the amount of memory to be set aside for the heap, specified as number of bytes (such as 65536), number of kilobytes (such as 128k) or number of megabytes (such as 1M). This argument is passed to CLDC.

The minimum heap required to run the `midp` executable when there are no applications installed on the emulator is 32k. (When applications are installed, the minimum amount varies but is never lower than 32k.)

The following command switches are supported:

`-autotest [-domain domain] descriptorURL`

Repeatedly installs, runs, and removes the first MIDlet from the MIDlet suite described by the Java application descriptor (JAD) file at the *descriptorURL* location. (The JAD file specifies the location of its corresponding JAR file.)

This command installs the MIDlet suite into *domain*. If *domain* is not provided the domain maximum is used. The command removes the MIDlet suite when it gets a 404 code from the server.

`-debugger [-port portNumber]`

Starts the MIDP executable in debug mode. The *portNumber* should be the number of the port on which the KVM debug proxy expects the debugger to be running. See the KVM documentation for more information on this and other options that are available for a MIDP executable in debug mode.

`-help`

Prints text to standard output that describes the options to the `midp` command and exits.

Note: Using the `-help` switch does not display the device skin.

`-install [-force] [-domain domain] [-removeRMS] descriptorURL`

Installs the MIDlet suite represented by the JAD file at *descriptorURL* location. This command installs the MIDlet suite into *domain*. If *domain* is not provided, and the suite is unsigned, the default domain, *untrusted*, is used.

Using `-removeRMS` removes any persistent data written by any previous versions of the MIDlet suite. This option only has an affect if the installation is overwriting an existing installation of the MIDlet suite.

Using `-force` has the MIDP Reference Implementation runtime ignore any overwrite errors that occur during the installation.

Note: Using the `-install` command-switch does not display the device skin.

`-list`

Prints information to standard output on the installed MIDlets, including their names, vendors, descriptions, storage names, sizes, original locations (URLs), and lists of MIDlets.

Note: Using the `-list` switch does not display the device skin.

`-remove (suitenumber | storagename | all)`

Removes the installed MIDlet suite identified by the given *suitenumber* or *storagename*, or removes all MIDlet suites.

Note: Using the `-remove` switch does not display the device skin.

`-run (suitenumber | storagename) [MIDletName]`

Runs the installed MIDlet suite that is identified by *suitenumber* or *storagename*. The *suitenumber* is the number displayed by the `-list` option to the `midp` command, and *storagename* is the name displayed by the `-storageNames` option to the `midp` command.

If *suitenumber* or *storagename* identifies a MIDlet suite that contains multiple MIDlets, *MIDletName* can specify a particular MIDlet in the suite to run.

`-storageNames`

Prints to standard output the storage names of the installed MIDlet suites. Storage names of MIDlets are implementation dependent.

Note: Using the `-storageNames` switch does not display the device skin.

`-transient [-force] [-domain domain] [-removeRMS]
descriptorURL [MIDletName]`

Downloads, installs, runs, and removes the MIDlet suite represented by the JAD file at the *descriptorURL* location. This command installs an unsigned MIDlet suite into *domain*. If *domain* is not provided for an unsigned MIDlet suite, the default, *untrusted*, is used.

If the JAD file at *descriptorURL* describes a MIDlet suite that contains multiple MIDlets, *MIDletName* can specify a particular MIDlet in the suite to run.

Using `-removeRMS` removes any persistent data written by any previous versions of the MIDlet suite. This option only has an affect if the installation is overwriting an existing installation of the MIDlet suite.

Using `-force` has the MIDP Reference Implementation runtime ignore any overwrite errors that occur during the installation.

`-version`

Returns the version of the MIDP Specification with which this release is compliant, the version of the MIDP implementation, and the version of the configuration (CLDC) with this release.

Note: Using the `-version` switch does not display the device skin.

Examples

Installing a MIDlet Suite

The following example installs the Games MIDlet suite from the URL <http://localhost:8080/midlets/games.jad>:

```
c:\midp2.0fcs> bin\midp -install
http://localhost:8080/midlets/games.jad
Storage name:
#Sun%0020#Microsystems%002c%0020#Inc%002e_#Sun#Samples%0020%00d
%0020#Games_
```

The following example installs the same unsigned MIDlet suite, but assigns it to the trusted domain:

```
c:\midp2.0fcs> bin\midp -install -domain trusted
http://localhost:8080/midlets/games.jad
Storage name:
#Sun%0020#Microsystems%002c%0020#Inc%002e_#Sun#Samples%0020%00d
%0020#Games_
```

Listing Installed MIDlet Suites

The following example lists the MIDlet suites installed on the device emulator. It shows the command being run on an emulator that has one MIDlet suite installed.

```
C:\midp2.0fcs>bin\midp -list
[1]
  Name: HttpView
  Vendor: Sun Microsystems, Inc.
  Version: 2.0
  Authorized by: CN=thehost;OU=JCT;O=dummy CA;L=Santa
    Clara;ST=CA;C=US
  Description: This midlet is ....
  Storage name:
    #Sun%0020#Microsystems%002c%0020#Inc%002e_#Http#View_
  Size: 17K
  Installed From: http://localhost:8080/midlets/pushdemo.jad
  MIDlets:
    HttpView
    HttpTest
```

Running an Installed MIDlet Suite

The following example runs the MIDlet suite listed in the previous example. It provides the number associated with the MIDlet.

```
c:\midp2.0fcs> bin\midp -run 1
```

See Also

[Appendix D, “The preverify Tool”](#)

[Chapter 2, “Using the midp Executable”](#)

The preverify Tool

`preverify` — prepare class files so that a MIDP implementation can run them. Optionally inspect the files for features of the Java™ programming language that are not part of the CLDC specification.

Synopsis

```
preverify [ -classpath path ] [ -d directory ] [ -cldc ] [ -nofinalize ]  
          [ -nonative ] [ -nofp ] [ @file ] [ -verbose ] inputFiles
```

Description

The `preverify` tool prepares class files, JAR files, and directory trees of class files so that they can be run by MIDP implementations. (The preverified class files can still be run by the Java 2 Platform, Standard Edition.)

The output of the tool mirrors the input: For each class file the tool is given, it writes a new class file of the same name to the destination directory. For each JAR file, it writes a new JAR file of the same name to the destination directory. For each directory name, it recursively handles every class file under that directory and writes new class files of the same names in the same hierarchical structures as the original directories.

In addition to creating new class files, the `preverify` tool can check the class files for Java programming language features that are not part of the CLDC specification. If a feature is not part of the CLDC specification, it cannot be used in code that MIDP will run. If it finds a forbidden feature, it reports the problem.

Options

The following options are supported:

none

Running the tool without options returns the same information as the `-help` option.

`-classpath path`

Specifies the location of the MIDP and CLDC classes. The *path* can contain directories or JAR files. On systems that run the Solaris™ Operating Environment, the list of locations is colon-separated. On Microsoft Windows platforms the list is semicolon separated.

`-d directory`

Specifies the destination directory where this tool writes its output files. The default directory is `./output`.

`-cldc`

Causes checks in the input class files for all Java programming language features that are not part of the CLDC Specification: floating point operations, finalizers, and native methods. It reports errors if it finds them.

Using this switch is equivalent to using the `-nofinalize`, `-nonative`, and `-nofp` switches.

`-nofinalize`

Causes checks for the use of finalizers in the input class files. It reports errors if it finds them.

`-nonative`

Causes checks for the use of native methods in the input class files. It reports errors if it finds them.

`-nofp`

Causes checks for the use of floating point operations in application classes. It reports errors if it finds them.

`-verbose`

Causes the printing of additional messages that describe the tool's actions. If this flag is not used and there are errors when creating an output JAR file, the errors will be written to *directory/jarlog.txt*. The *directory/jarlog.txt* file will not be created if *inputFiles* does not contain a JAR file or if there are no errors.

@fileName

Specifies the name of a text file from which the options and operands to the tool will be read. The parameters must be on a single line. If `-classpath` or `-d` are present, their values (*path* and *directory*) must be enclosed in double quotes.

For example, the contents of *fileName* could be:

```
-classpath "api/classes;e:/samples/classes" -d "output" -nofp
-verbose HelloWorld1 HelloWorld2 HelloWorld3
```

Operands

The following operand is supported:

inputFiles

One or more files to be preverified. The files can be in three different formats:

- Java class files
- Directories containing Java class files
- JAR files containing Java class files

The names of the input files should be separated by white space.

Exit Status

0

Successful completion.

>0

An error occurred

Examples

The following examples assume that the directory that holds the `preverify` command is a value on your `PATH` environment variable.

Verifying Class Files

The following command preverifies class files but does not check them for features that are not in the CLDC specification. It writes the output to the

`c:\work\myMIDlet\classes` directory.

```
c:\work\myMIDlet\tmpclasses> preverify -d c:/work/myMIDlet/classes
                             -classpath c:/midp2.0fcs/classes MyMIDlet
                             alerts/About forms/Splash
```

The following command is a simpler version of the preceding one. Instead of listing the class files individually, it specifies the parent directory of the class files.

```
c:\work\myMIDlet> preverify -d c:/work/myMIDlet/classes
                             -classpath c:/midp2.0fcs/classes tmpclasses
```

Verifying and Checking Files

The following example preverifies Java class files and checks them for the features that are not in the CLDC specification. Its output will go into the default location, the `./output/` directory.

```
c:\work> preverify -classpath c:/midp2.0fcs/classes -cldc
myMIDlet/tmpclasses MIDletFromCoworker.jar AnotherClass
```

The following example preverifies Java class files and checks them for floating point operations. It assumes that the MIDP class files are in the `CLASSPATH` environment variable, and writes the output to `c:\work\myMIDlet\classes` directory:

```
c:\work> preverify -d c:/work/myMIDlet/classes -nofp
myMIDlet/tmpclasses/MyMIDlet MIDletFromCoworker.jar
```


Environment Variables

The following environment variables interact with specific features of this tool:

PATH

Provides the location of the `jar` tool, which this tool calls to create the new, output JAR file. The `jar` tool must be accessible on your `PATH` if you provide a JAR file as input to the `preverify` tool.

CLASSPATH

Provides this tool with a path to MIDP, CLDC, and user-defined classes. The `-classpath` switch overrides the value of this environment variable.

See Also

Creating MIDlet Suites

Index

C

Certificate Authorities, 34, 41
Certificate Chain, 34

D

Device Emulator
 Navigation Buttons, 5
 Overview, 4
 Skin, 4
 Starting, 6
Device Simulation, 26

I

Importing a Key, 44

J

JAD File, 49
 Adding a Certificate, 52
JadTool Utility, 49
javax.microedition.io.PushRegistry
 permission, 35
JCA Keystores, 53

K

Keys
 Adding to ME Keystore, 44
 Deleting, 46
 Listing, 45
 Replacing, 46

M

ME Keystores, 53

Creating Alternates, 43
Managing Alternate, 43
Working with Multiple, 43

MEKeyTool, 42, 53

 Adding a Key, 55
 Deleting a Key, 55
 Listing Keys, 53, 55
 Options, 54

Midlet

 Running, 14

Midlet Suite

 Changing Permission Levels, 20
 Getting Information, 16
 Installing, 7
 Listing, 30
 Protection Domains, 37
 Push Functionality, 12
 Removing, 17
 Running, 14, 64
 Updating, 19

Midlet Suites

 Installing, 57, 63
 Listing, 63, 64

Midlets

 Trusted and Untrusted, 33

MIDP

 Alternate Keystores, 44
 Device Simulation, 26
 Executable, 25
 Handling Certificates, 47
 midp Command, 26, 57
 Reference Implementation, 3, 23
 Running, 44
 Security Features, 33

O

Options, 66

P

Permission Requests

 Midlet or Midlet Suite, 15

Permissions, 33

permissions, 35

preverify Tool, 65

 Environment Variables, 69

 Options, 66

 Verifying and Checking Files, 68

 Verifying Class Files, 66, 68

Property Values

 Changing, 27

Protection Domains, 35

Public Keys, 41

 MEKeyTool, 42

push functionality, 35

PushRegistry class, 35

R

Root Certificate, 34

Running a Midlet or Midlet Suite, 14

S

Signing a JAR File, 52

V

Viewing a Certificate, 52