Java ME Permissions

Version 0.9, Draft

INFORMATION GUIDE

COPYRIGHT

Trademarks and Service Marks

About This Document

This document gives description of the security features in MIDP platform and various permissions associated with various APIs.

Scope:

This article assumes user to be familiar with Java programming and the basics of MIDP programming.

Document History:

Date	Version	Comment
05/02/09	0.9	Draft

References:

1. Understanding MIDP 2.0 is Security Architecture:

http://developers.sun.com/mobility/midp/articles/permissions/

Abbreviations:

JRE	Java Runtime Environment
JTWI	Java Technology for Wireless Information
API	Application Programming Interface
VM	Virtual Machine
CDC	Connected Device Configuration
CLDC	Connection Limited Device Configuration
MIDP	Mobile Information Device Profile
JSR	Java Specification Request
WMA	Wireless Messaging API
MMAPI	Mobile Media API

Table of Contents

Table of Figures

List of Tables

Introduction

MIDP has always been a good platform for mobile users and the main feature of MIDP is its focus on security. As more numbers of JSRs are being introduced every year, Java ME applications are offering developers greater access to various data and services without compromising on the MIDP security system.

The security feature in MIDP governs how data and services are being accessed. Security to the Java ME Applications is achieved by providing permissions to protected APIs or functions.

Providing greater access to data and services on a device requires a level of trust to be established between the application, the device, and the user.

Overview

Permissions are named similar to Java package names. These are used to protect APIs or functions that are sensitive and require authorization.

Permissions names are case sensitive. All of the permissions for an API must use the same name as that of the API. If the permission is for a function of a specific class in the package then the permission **MUST** include the package and class name.

For example: MIDlet that needs to make a socket connection would need the permission of *javax.microedition.io.Connector.socket*.

Security in MIDP

In MIDP 1.0, security is mostly achieved by removing the ability to perform sensitive operations.

MIDP 2.0 comes with enhanced security features that open up more capabilities of devices for the developers without compromising security.

Security implementation checks whether a MIDlet Suite has the necessary permission before invoking the protected APIs or functions. MIDP 2.0 accomplishes this using protection domain.

Protection Domain

A collection of Permissions, which can be granted to a MIDlet suite is called protection domain.
Trusted MIDlet suite contains the permissions that authorize access to protected APIs or functions. A Signed MIDlet Suite is bound to protection domain and treats MIDlet suite as Trusted.

Untrusted MIDlet suites execute in the restricted environment with security constraints. The access to protected APIs or functions either is not allowed or is allowed with explicit user permission. This is because origin and authencity of the MIDlet suite cannot be determined and guaranteed.

A protection domain consists of:

- Allowed Permissions
- User Permissions

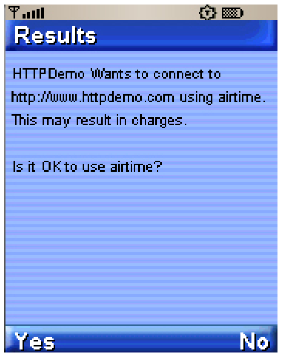
Allowed permissions:

Allowed permissions is a set of permissions that should be allowed (granted to contained MIDlet suites). These permissions allow access to a given protected API or function. Allowed permissions do not require any user interaction, so user is not prompted for granting access to the API or functions.

User permissions:

User permissions is a set of permissions that the user may need to authorize. These permissions prompt the user for granting access to protected APIs or functions. Granting access to some protected APIs or functions may lead to air time charges. For example: Http Connection, Wireless Messaging API.

Figure 1 shows the permission prompt shown to the user and the airtime charges indication.

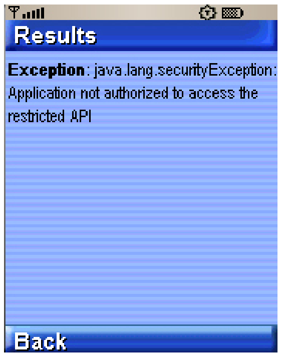


HTTPDemo Wants to connect to http://www.httpdemo.com using airtime. This may result in charges.

Is it OK to use airtime?

Figure 1: Permission Prompt

Figure 2 shows the security exception thrown to the user when permission is denied. Only on user confirmation, the access to the API or function is given; otherwise, *java.lang.SecurityException* is thrown on denying the permission.



```
Exception: java.lang.securityException: Application not authorized to access the restricted API
```

Figure 2: Security Exception

Following are the various user interaction modes defined to allow or deny permissions. These user permissions vary as per the security policy.

Blanket

Blanket is valid till the MIDlets remain installed on the device or user changes the permission. In this mode, the user permission is valid for every invocation of an API or function.

Session

In this mode, user permission is valid till the MIDlet suite terminates. User is prompted on or before the first invocation of the API or function, which is protected. The permission is prompted once for protected API invocation or function. The process is repeated again when the MIDlet suite starts execution.

Oneshot

In this mode, the user is prompted for permission on each invocation of the protected API or function. The user permission is taken each time the MIDlet suite invokes protected API or function.

Requesting MIDlet-permissions

MIDlet suite may request permissions to access protected APIs or functions. Requesting MIDlet permissions is achieved by declaring permission request in MIDlet application descriptor file. The attributes are *MIDlet-Permissions or MIDlet-Permissions-Opt*.

If the permissions are requested using *MIDlet-Permissions*, then MIDlet suite installation might be aborted, if the MIDlet cannot gain access to the requested API.

If the permissions are requested using *MIDlet-Permissions-Opt*, then MIDlet suite installation can be continued, even if the MIDlet is unable to gain access to the requested API.

Multiple permissions can be specified and are separated by commas.

```
MIDlet-Permissions: javax.microedition.io.Connector.http, javax.microedition.io.PushRegistry

MIDlet-Permissions-Opt: javax.microedition.io.Connector.https
```

Adding permissions in MIDlet

Following steps shows how to add permissions to the Midlet.

1. Open Samsung SDK.
2. Open Midlet Project.
3. Click on the Settings Menu from the Samsung SDK menu bar as shown in Figure 3.

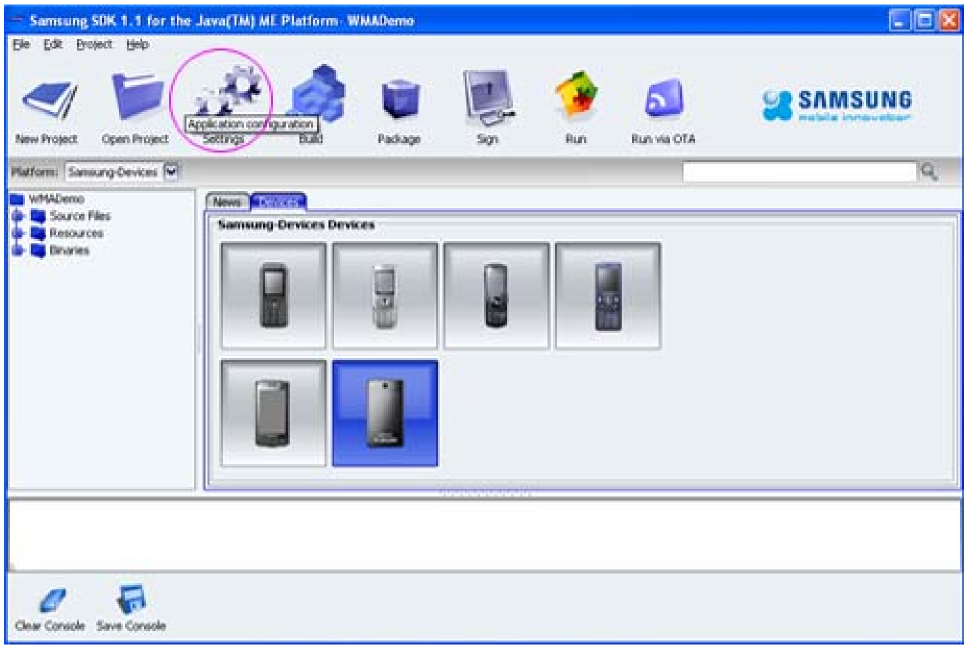
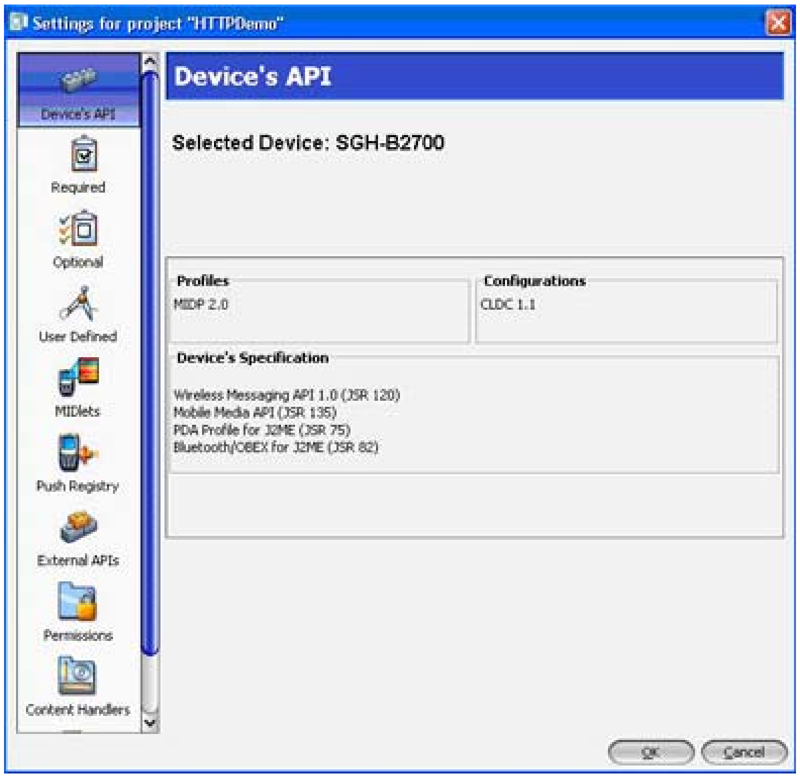


Figure 3: Samsung SDK window

4. Settings dialog appears as shown in Figure 4.



Device's API

Selected Device: SGH-B2700

Profiles: MIDP 2.0

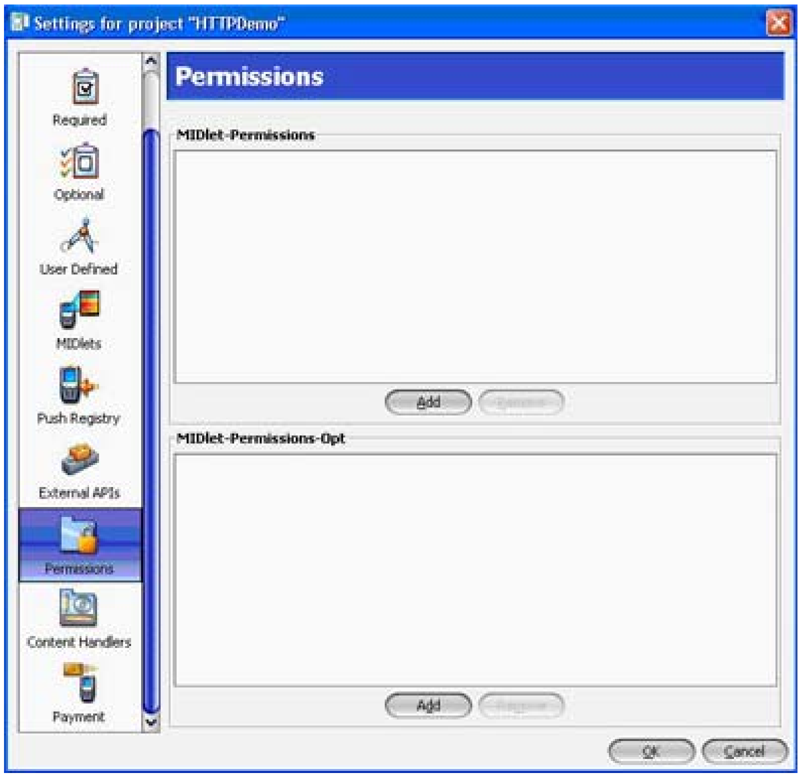
Configurations: CLDC 1.1

Device's Specification

Wireless Messaging API 1.0 (JSR 120)
Mobile Media API (JSR 135)
PDA Profile for J2ME (JSR 75)
Bluetooth/OBEX for J2ME (JSR 82)

Figure 4: Settings dialog

5. Click on the Permissions Option. It will open up Permissions dialog as shown in Figure 5.



Permissions

MIDlet-Permissions

MIDlet-Permissions-Opt

Figure 5: Permissions

6. Click on the "MIDlet-Permissions" Add button to add permissions. These permissions would be set to the MIDlet-Permissions attribute in JAD file. On clicking Add button, API Selection Dialog appears as shown in Figure 6.

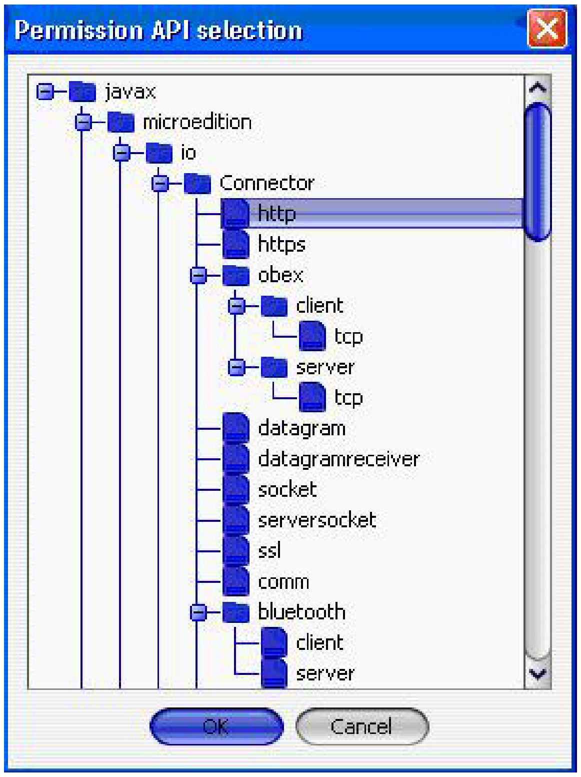
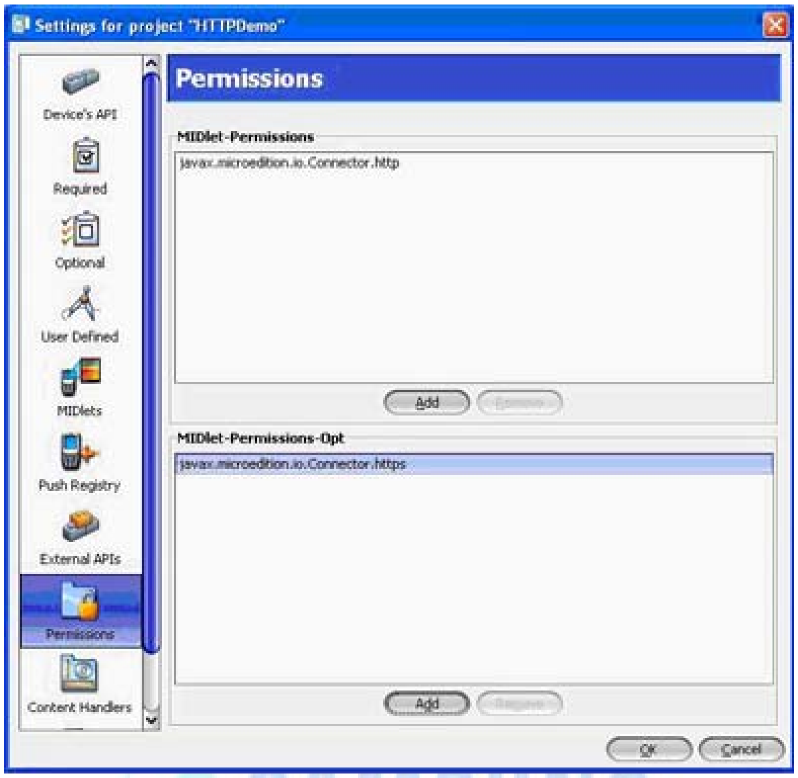


Figure 6: API selection dialog box

For example: For Http Connection, permission for *javax.microedition.io.Connector.http* can be requested.

7. Click on the "MIDlet-Permissions-Opt" Add button to add optional permissions. These permissions would be set to the MIDlet-Permissions-Opt attribute in JAD file. The procedure to add permission is same as the one shown in step 6. For example: *javax.microedition.io.Connector.https* connection can be requested.

8. On adding permissions the permissions dialog box looks as shown in Figure 7



MIDlet-Permissions
javax.microedition.io.Connector.http

MIDlet-Permissions-Opt
javax.microedition.io.Connector.https

Figure 7: Permissions dialog box

The requested permissions are reflected in the JAD file as:

```
MIDlet-Permissions: javax.microedition.io.Connector.http
MIDlet-Permissions-Opt: javax.microedition.io.Connector.https
```

Detecting Permissions in MIDlet

The MIDlet class in MIDP 2.0 includes a method

```
int javax.microedition.midlet.MIDlet.checkPermission(String permissions)
```

That allows passing the permission name as a parameter. An integer value is returned indicating the status of the permission.
Where status can be:

0: if the permission is denied.

1: if the permission is allowed.

-1: if the permission status is unknown.

Following table shows some API list, which requires permission for invocation or its protected function.

Table 1: Permission Table

Network Generic Connection Framework	javax.microedition.io.Connector.http javax.microedition.io.Connector.https javax.microedition.io.Connector.cbs javax.microedition.io.Connector.datagram javax.microedition.io.Connector.datagramreceiver javax.microedition.io.Connector.socket javax.microedition.io.Connector.ssl javax.microedition.io.Connector.serversocket
Messaging JSR 120	javax.microedition.io.Connector.sms javax.microedition.io.Connector.mms javax.wireless.messaging.cbs.receive javax.wireless.messaging.mms.receive javax.wireless.messaging.mms.send javax.wireless.messaging.sms.receive javax.wireless.messaging.sms.send
Auto invocation MIDP	javax.microedition.io.PushRegistry
Bluetooth JSR 82	javax.microedition.io.Connector.comm javax.microedition.io.Connector.bluetooth.client javax.microedition.io.Connector.bluetooth.server javax.microedition.io.Connector.obex.client javax.microedition.io.Connector.obex.client.tcp javax.microedition.io.Connector.obex.server javax.microedition.io.Connector.obex.server.tcp
File read , write data access JSR 75	javax.microedition.io.Connector.file.read javax.microedition.pim.ContactList.read javax.microedition.pim.EventList.read javax.microedition.pim.ToDoList.read javax.microedition.io.Connector.file.write javax.microedition.pim.ContactList.write javax.microedition.pim.EventList.write javax.microedition.pim.ToDoList.write
Location JSR 179	javax.microedition.location.Location javax.microedition.location.Orientation javax.microedition.location.ProximityListener javax.microedition.location.LandmarkStore.category javax.microedition.location.LandmarkStore.management javax.microedition.location.LandmarkStore.read javax.microedition.location.LandmarkStore.write