Java ME FAQ

This article contains the most frequently asked questions on the Java Discussion Boards. Many important pieces of information that can help you solve the most basic/recurrent problems when developing in Java ME.

I cannot get this midlet of mine working? Please HELP ME! Urgent!

First check the Javadocs for that API (those are included in all SDK installations). There are also sample midlets and documentation available on Nokia Developer website ([1]). And you can find wealth of information in here (Discussion Boards at [2]) and on Sun's JavaME web site ([3]).

If you are still having problems, please include relevant parts of the source code in your post, and also specify which devices you are working on. If the problem has something to do with your desktop computer, please add OS information for your desktop computer too. In some cases the firmware version of the device might provide valuable information to the readers (*#0000# in the idle screen).

What tools do I need to create my first midlet?

You can write your code in your favorite text editor or you could use an IDE (Integrated Development Environment), such as Eclipse, Netbeans, or JBuilder. [Note: reference to Carbide.j has been removed as Nokia does not provide that tool anymore]

When you use Sun's Wireless Toolkit, NetBeans with Mobility pack, and other mobile Java development environments, you should remember that if you are developing your midlet for Nokia devices, it is advisable to test the application with Nokia emulators (as well as on real devices), as the generic emulators behave differently from Nokia emulators and Nokia devices.

Nokia Asha SDK 1.0 (beta) (14 May 2013): Nokia Asha SDK 1.0 (beta) is available for download. This SDK enables you to target your Java apps at phones based on Nokia Asha software platform 1.0. SDKs for Series 40 (including *Nokia SDK 2.0 for Java*), can be downloaded from here.

I need to download Nokia Development Suite X.X for J2ME. Where can I find it?

NDS was replaced with Carbide.j, which was in its turn removed from Nokia's tools portfolio in 2007. Please use Eclipse + EclipseME or NetBeans + Mobility Pack for your MIDlet development. See Getting started with Java ME and Installing Java ME development tools for S60.

How can I transfer my midlet to the handset?

On Symbian phones and newer Series 40 phones: Send the JAR and JAD files over Bluetooth from the File Browser.

On older Series 40 phones, you need to use PC Suite. PC Suite works well also with Symbian and newer Series 40 phones.

An advanced option is to store the JAR and JAD files on a Web server and connect to the JAD file using the phone browser. Make sure the mime types are set correctly on the server ("application/java-archive" for jar and "text/vnd.sun.j2me.app-descriptor" for jad).

How can I debug my midlet?

All SDKs contain an emulator, which basically emulates the phone Java environment and the set of APIs included in that SDK on your PC. When you run your midlet in the emulator, you can see diagnostic information (like printStackTrace() output) in the IDE's console screen.

You should always test your application on a real device as some of the phone capabilities cannot be emulated on PC. Also there are subtle differences in the phone implementations.

Why are there so many SDKs, can't I just use one to code and test my midlet?

Nokia Asha SDK 1.0 (beta) (14 May 2013): Nokia Asha SDK 1.0 (beta) is available for download. This SDK enables you

to target your Java apps at phones based on Nokia Asha software platform 1.0. SDKs for Series 40 (including *Nokia SDK 2.0 for Java*), can be downloaded from here.

Different phones have different set of APIs, and the SDKs provide a compatible set of APIs for each device (and an emulator to test the midlets on). Nokia devices are based on platforms (like Series 40, Symbian , and Series 80), editions (1st, 2nd, and 3rd edition so far), and Feature Packs. Each device belongs to one of platform edition / feature pack combination.

For example if you are developing an application for N70 smart phone, which is a S60 2nd Edition Feature Pack 3 device, you should be using that specific SDK to develop the midlet on. If you are using an earlier SDK version, some of the APIs might be missing. If you are using a newer SDK version, you might end up using features which are not available on N70. Additionally the emulator of the S60 2nd Ed FP3 SDK provides the closest emulation of that specific device of all the available emulators.

Does my handset support API xyz?

Please check the device specifications at http://www.developer.nokia.com/Devices/

You can also use a shortcut URL – add the phone model number in the end of the previous URL. For example if you want to see the specifications of Nokia 6131 phone you can go to this URL
http://www.developer.nokia.com/Devices/Device_specifications/6131/

Or if you are looking for API availability on devices from other manufacturers, you can also check the J2ME Polish Device Database at [4].

For most of the newer APIs it is also possible to query the phone during execution time if a certain API exists. This is done using System.getProperty() method call. Please check the correct query strings (for example microedition.pim.version for PIM API) from this document [5]

A special case is MMAPI, which contains a lot of optional features. For example not all phones are able to use the camera. You can check how the MMAPI is supported on various Nokia devices from this doc MMAPI Support In Nokia Devices

My phone does not seem to have support for API xyz. Can it be added on the phone?

Short answer: No.

Longer answer: No, but if the API can be implemented on using the existing Java APIs on the phone, you can include the required class files in your midlet. This of course makes your midlet bigger and if you need to use the same classes in another midlet, you need to include the classes in that midlet too, as the classes of another midlet are not accessible from other midlets. Also note that in most cases this is not possible as usually the new API requires access to the system functions of the phone.

Can I read and write files on my mobile phone using a midlet?

Yes, if your phone supports the FileConnection API, which is an optional package of JSR-75. Check also this introductory document and the example midlet included here

Can I access the address book /to-do list/calendar on my phone

Yes, if your phone supports the PIM API, which is an optional package of JSR-75. Check also this introductory document and the example midlet included here

My midlet seems to be too big for my phone. How can I make my JAR files smaller?

Obfuscation replaces the class and method names with short (1 character) names, which has the side effect of making the class files smaller. Originally obfuscation was intended to be a tool to make the reverse engineering of the midlet harder. One example of such tools is Proguard.

If your application uses resource files (images, audio, or video), make sure they are optimized for the small devices (suitable bit depth, image resolution, audio bit rate, and so on.)

You can also shave some bytes off your JAR-file by making the folder names really short.

Every time my MIDlet tries to access network/contacts/... there are these annoying confirmation dialogs. Can I get rid of them?

The new security model introduced in MIDP 2.0 is protects the phone and the user from malicious applications by restricting access to APIs that are considered sensitive. In general the midlet is granted only minimal access to these APIs, meaning that the user is asked for confirmation every time the API is accessed.

The user can manually change the default permissions a little through the Application Manager (on Symbian devices use the separate Application Manager; on Series 40 devices you can change the application settings from the through the Application Menu).

The developer can also sign the MIDlet with a certificate so the MIDlet has less restricted access to the sensitive APIs. The corresponding root certificate has to be available on the target phone; otherwise the installation will fail. The Java Verified certificate is widely available on devices from all manufacturers. Your MIDlet will be signed with this certificate after it passes the Java Verified testing. The testing criteria is available at Java Verified Web site (http://javaverified.com/). For more information on signed MIDlets, see Signed MIDlet Developer's Guide ([6]).

What kinds of permissions are available for signed and unsigned MIDlets?

This MIDP 2.0 specification addendum ([7]) lists the default permissions as well as other available permissions for trusted 3rd party MIDlets and untrusted MIDlets. Note that some carriers/operators grant different permissions to the MIDlets. For example Cingular's Java Signing Overview documentation is available on the at&t developer Web site (http://developer.att.com/developer/forward.jsp?passedItemId=300004m).

What certificates should I use to sign my MIDlet?

You can sign your MIDlet with any certificate available on the target devices and allowing the Java Application installation. (You can check what certificates are available on the device through security settings – make sure the certificate you are thinking to use is set to allow Java application signing).

The Java Verified certificate is widely available on devices from all manufacturers. Your MIDlet will be signed with this certificate after it passes the Java Verified testing. The testing criteria is available at Java Verified Web site (http://javaverified.com/).

Can I use a certificate and key created by me to sign my midlets?

Yes, you can run a midlet signed with a certificate created by you on emulators. Self-signing does not work on Series 40 devices nor S60 3rd edition devices.

I have a signed MIDlet I cannot install on my device. Is there any way to install this MIDlet?

Check that all the required permissions are requested in the JAD file. You can also try removing the *MIDlet-Jar-RSA-SHA1:* and *MIDlet-Certificate-1-1:* entries from the JAD file (your signed MIDlet will now be treated as unsigned midlet). If the installation still fails, check that the JAR file size attribute is correct in the JAD file. Make also sure that all the JAD properties are set a non-empty value.

How do I retrieve the phone number from my midlet?

You would have to use the JSR 253 - Mobile Telephony API which allows you to have a fairly good set of call-related functionality, so it's possible that this API will also provide a way to let you know what's your phones's number. No Nokia devices currently support JSR 253.

There are some possible workarounds:

- When the user first downloads the application, have a server-side app such as a Java servlet or PHP script that retrieves the phone number from the HTTP header, and set it as a property in a dynamically-generated JAD file. This way, when your midlet is running, it can check the phone number using `getAppProperty()` method from MIDlet class. This approach will only work if the device is accessing the server-side application using a WAP access point, however. This happens because in this type of access point, an HTTP header containing the MSISDN (phone number) is added to the HTTP request that will be forwarded to your application, so you can retrieve it.
- If the target handset supports the Wireless Messaging API, you can have your application register an SMS connection, using PushRegistry, with a non-standard port. When the user first starts the application, generate some random data and send it to the phone number the user will input on a text field. If the phone number is correct, than you'll receive the message in a few

seconds on the connection you've registered. You can then check the data to see if it matches the data you generated earlier, and if it does, get the sender phone number using [CODE]Message.getAddress()[/CODE] method and save it to RMS. Note that the default security setting for Wireless Messaging API is "Ask Always", so this will cause security prompts if the midlet is not signed and/or the application security settings are not configured correctly.

- If you are using an Symbian device, you can write a Symbian C++ application that retrieves the phone number and opens a server socket on 127.0.0.1. The midlet then connects to this localhost socket and retrieves that information. There are some drawbacks to this approach: Your C++ application has to be started on boot, the method used to retrieve the phone number is not portable (if you're getting only the IMEI code, not the phone number, then it's portable); third, it depends on a C++ application, which greatly increases complexity by itself.

How do I get the IMEI of a phone from my midlet?

Refer to How to get IMEI in Java ME

Will my Java ME application run unmodified across all platforms?

Nokia adopts a platform approach on its products. It means that if you write an application for a given platform, it is supposed to run mostly unmodified on all other devices from the same platform. Currently there are three platforms: Series 40, Symbian and Series 80.

There are several degrees of portability, however:

- If you use only MIDP 1.0 features and LCDUI components (excluding Canvas), your application will run unmodified on all Nokia devices which support Java.
- If you use only MIDP 2.0 features and LCDUI components (excluding Canvas), your application will run unmodified on all Nokia devices which support MIDP 2.0 profile.
- If you use some of the optional APIs, you'll have to adapt your application according to one of the cases below:

- **Source code-level differences**: Those are found when you are using a certain API package (such as FileConnection or Bluetooth) that is supported in a device or platform, but not in other devices. For example, Bluetooth API (JSR 82) is supported on Nokia 6230i, but not on Nokia 6101. To overcome these differences, you'll have to produce different versions of your midlet for each handset (or platform), according to the availability of the given API. In order to make this easier, there are several software packages that can help you produce multiple versions of your source code; two of the most popular use C++-like pre-processor macros:
  - Antenna
  - NetBeans.
- **Runtime-level differences**: Those are found when you're using a certain API package that has runtime optional components. This happens mainly in Mobile Media API, which has a large set of optional features that may or may not be implemented by a device, but if those are indeed implemented, there are no changes in the source code, only in runtime behavior. For example, both 6230 and 6681 support Mobile Media API, therefore there are no source code-level differences here. However, 6230 does not support capturing pictures with the device's camera, neither capturing and recording audio, whilst 6681 supports both operations. In order to make your application run in both devices, you'll need to perform runtime checks, usually using system properties, to provide the best behavior for a good user experience.
- **Hardware-level differences**: Those refer mostly to the screen size and resolution. For example, 6230i has a 208x208 display, while E61 has a 320x240 landscape-oriented display. As I said before, if you're using only LCDUI classes, you don't need to worry about that, but if you are using screen size-dependent graphics, you'll have to either have multiple versions of your midlet or to perform runtime checks and adjusting your drawing functions according to the screen size.

Can I acces DRM-protected content using Mobile Media API?

On S60 3rd. Edition devices, there is some DRM support. You can play a DRM-protected file, using the following syntax:

```
Player p = Manager.createPlayer("file:///C:/Path/To/File.dcf");
p.start();
```

I signed my midlet with a Verisign/Thawte and it's not installing on device *X*, what's wrong with it?

The root certificate for your certificate is either 1) not present in the root CA store of the device 2) not enabled to authorize and authenticate software installation.

You can check whether this is the case this way:

On Series 40 devices, go to "Menu/Services/Settings/Security Settings/Authority certificates/Certificate list". Check if the root certificate is there and, if so, it's enabled for application authentication, choosing "Options/Select use". Older Series 40 devices have the certificates under Services.

On Symbian devices, go to "Tools/Settings/Security/Certificate management/Authority". To check if a given certificate is enabled for application authentication, go to "Options/Trust Settings".

I signed my midlet and I'm still getting the security prompts when trying to access protected APIs.

First, you need to check if your application has the proper permissions for the operations you're trying to perform:

- On Series 40, go to "Menu/Applications/Collection/Select Application/". Highlight your application's name, go to "Options/Application access", and configure the proper permissions for the actions your app wants to perform.
- On Symbian, go to "Tools/App. Manager/". Highlight your application's name, go to "Options/Suite Settings", and configure the proper permissions for the actions your app wants to perform.

Please note that the default access rights are not ever the same as max access rights. The user has to manually change them.

If the permissions are correctly configured and you still get the security prompts, it may be that your phone has a firmware version customized for a given operator, which has the option of customizing the security domains policy, not allowing, for example, a midlet to connect to the network without user confirmation, or its access to the filesystem, using FileConnection API. This is unusual behavior, because in most cases like this the installation will simply fail.

In that case, you have to contact your operator, and ask them about their developer programs and partnering agreements, usually available only for companies, not individuals. Partnering with the operator makes it possible to sign your midlet with their certificate, which would put your midlet in the "Operator" security domain, giving it looser permission settings.

Why can't I have my midlet auto-started with PushRegistry without a security prompt being shown, even though the application is signed?

See Variance in security domains for MIDlets on certain operator variant Series 40 2nd Edition phones (Known Issue).