Tips for debugging MIDlet startup issues

This article explains how to debug problems when you attempt to run your Java ME MIDlet on a real device. It covers both symptoms and likely causes.

Cannot Even Download!

Symptom: You are trying to install over the air (OTA, from an HTTP server), and you get a message like "unknown file type", or the JAD file displays in the browser as a text file, then your server might not have the correct MIME types configured.

A "MIME type" is the piece of information sent with a file, that enables the receiver (like the browser) to know what kind of file it is, and process it correctly. Servers usually work out the MIME type by looking at the file name's extension, and looking that up in their configuration file.

The types you need are:

.jad	text/vnd.sun.j2me.app-descriptor
.jar	application/java-archive

You may need to ask your server's administrator to add these for you.

Some devices do not like certain characters in the file names. Keep to printable ASCII characters (those with codes 33-126). Avoid characters that are often forbidden in file names, like "*", "?", etc. Avoid spaces. Keep to "a"-"z", "A"-"Z" and "0"-"9". Ideally, keep to lower case, to avoid problems between case-sensitive (Unix/Linux) and case-insensitive (Windows) file systems. Underscores ("_") are usually OK.

Avoid having more than one "." in the name. Some devices get confused with names like "myapplication.1.0.jar", seeing the file type as "1.0.jar" instead of "jar", and thinking it is not an application.

Symptom: You see a message like "JAR too large" or "invalid JAR" when attempting to download, yet there is more than enough space on the device.

Some devices have maximum file size limits. JAD files should ideally be kept below 5k. JAR files may have device-specific limits. Series 60 devices do not impose JAR size limits, but Series 40 devices do (as do some devices from other manufacturers). Check the device specifications for the maximum JAR size.

Check what is in your JAR... sometimes over-sized JARs are caused by unwanted files creeping into the build process. Source code files and "thumbs.db" are common culprits.

Symptom: During an OTA download, the download stops prematurely, and the application is not installed.

Provided you install a JAD first, the phone knows how big the JAR will be before downloading it, and should not download a JAR it cannot install. While some devices will allow a JAR-only installation, this is not recommended. Use a JAD, ensuring that the MIDlet-Jar-Size is correct.

Network problems can also cause premature download termination. Typical problems are:

- Loss of signal - try downloading somewhere where the signal is stronger
- Switching between 3G and 2G - if you have a good 2G signal but a poor 3G signal, try disabling 3G on the device (usually there is an option in phone network settings, set to "GSM" rather than "auto")
- Network download limitations - some networks may limit download sizes, so try downloading something smaller (say, less than 64k, and work up to establish the limit)

Sometimes, network limits appear to be specific to a cell, so you may find yourself unable to download a file, while someone in another city, using the same network, can.

Problems Installing by Bluetooth or Infrared on Series 60

Symptom: Installation fails when installing from the message inbox (non-OTA installation).

Send JAD and JAR files to the Series 60 message inbox, and installing from there, is a frequent source of problems. Common messages refer to "JAD/JAR mismatch" or "Version already installed". Problems may also occur with signed builds (they appear to be unsigned, or to have invalid signatures).

The JAD and JAR will appear in the inbox as two separate items. Since you can only select one to install, the installation process searches the inbox for the other file. Sometimes, it may find a file from a previous installation (which will not match, or will match a version already installed).

Try emptying the inbox, re-sending the JAD and JAR, and installing again.

After installing an application, delete it from the inbox (both items, if you used a JAD as well as a JAR).

Invalid JAR File

Symptom: The JAR downloads OK, but the devices reports "invalid JAR file" or "invalid application", either immediately, or when you try to launch the application. Other symptoms might include "unable to start application" or "java.lang.ClassNotFoundException".

First thing to check is the /META-INF/MANIFEST.MF file in the JAR. If you are installing a JAD, you will need to check that too. These files are plain text files, and can be viewed in Notepad. Remember to check the *actual* MANIFEST file from the JAR. Extract it using the command-line JAR utility, or WinZip. These files must contain:

Attribute	Value	MANIFEST?	JAD?
MIDlet-Name:	*name*	Required	Required
MIDlet-Version:	*major.minor.micro*	Required	Required
MIDlet-Vendor:	*name*	Required	Required
MIDlet-1:	*name*, *icon-file-name*, *class-name*	Required	(optional)
MicroEdition-Profile:	MIDP-*x.x*	Required	(optional)
MicroEdition-Configuration:	CLDC-*x.x*	Required	(optional)
MIDlet-Jar-URL:	*relative-url*	Never	Required
MIDlet-Jar-Size:	*size of JAR in bytes*	Never	Required

ClassNotFoundException may indicate that the class specified in MIDlet-1 is not present in the JAR. Remember to check *what is actually in the JAR*, not what you think is there.

Notes:

- Many devices will accept different version number formats, but some will require a three-part version, like "1.0.1".
- Some devices do not like version numbers with many digits in each part. Try reducing the three parts to just one digit each, like "1.2.3".
- The *icon-file-name* must refer to a PNG file, in the JAR, with the full path name (starting with a "/"). If there is no icon, the file name can be omitted. In this case, both commas are still needed.
- The *class-name* must refer to a class in the JAR, with the fully qualified name (including any package name), and that class must be a public class that extends javax.microedition.midlet.MIDlet.
- The MIDP and CLDC versions specified *must* be compatible with the device. Versions are backwards compatible, so a CLDC-1.1 device, for example, is always compatible with CLDC-1.0. A CLDC-1.0 device might refuse to run a CLDC-1.1 application.
- The JAR URL can be relative. That is, if the JAR is in the same folder on the same server as the JAD, you need only the JAR file name.
- The JAR size *must* match the size of the JAR file exactly.

Where the same attribute appears in both JAD and Manifest, the value in each *must* be identical.

Sometimes you get JAD/Manifest mismatch problems, even though they *do* match. This can happen when the device or the network's HTTP proxy server caches an older version, when you're downloading OTA. Look for the browser's option for clearing its cache. If that doesn't help, try uploading the JAD and JAR again, but with different file names.

Example Manifest:

```
MIDlet-1: My Application, , com.me.MyMidlet
MIDlet-Name: My Application
MIDlet-Version: 0.1.0
MIDlet-Vendor: me
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
```

Users of eclipseME need to configure these in the Application Descriptor Editor. There is corresponding documentation for the Eclipse MTJ, which replaces eclipseME.

Symptom: You see a message about "incompatible" or "unsupported version" (or may be reported as "invalid JAR" or "invalid application").

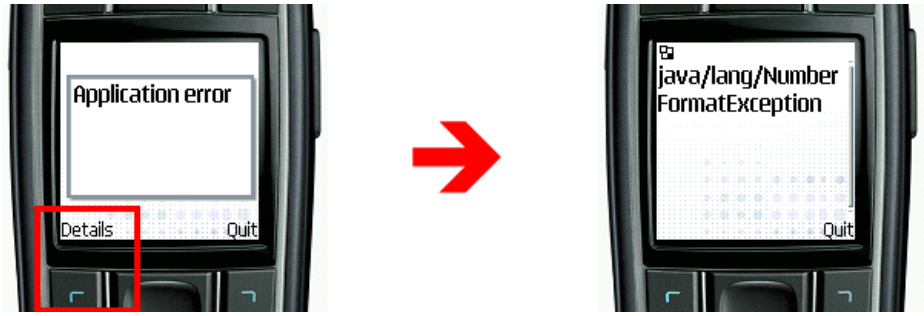
A particularly common problem is where IDEs have automatically inserted "CLDC-1.1" and "MIDP-2.1" as the configuration and profile versions mentioned above. Try adjusting these to match your device (check the device specifications).

Configuration and profile versions should be set to match the requirements of your application.

Application Error

Symptom: Device displays "Application Error" when the MIDlet is launched.

This indicates that an exception has been thrown, and not caught by the application. On some devices, you will be able to find out the name of the exception by selecting "Details".



Possible causes:

java/lang/OutOfMemoryError	The emulator was providing more memory to your application than the device can. Check the heap size specification for your device, and reduce the amount of memory your application needs accordingly.
java/lang/NoClassDefFoundError	A class required by your application is missing. This may be because you are using an API that is not present on the device. Check the device's specification. For example, you cannot use Bluetooth classes and methods on a device that does not support JSR-82. If you are using a third-party library (such as kXML), your build process will need to copy the class files from the library JAR into your application's JAR, so that they are present on the device at run-time.
java/lang/NullPointerException	Something is not being initialized correctly. Check the start-up process of your application. Are you catching and ignoring exceptions thrown, for example, by createImage()? If so, you may be hiding the real problem, but leaving some variables set to null and causing this problem later in the code.

The phrase "bad or missing stack map" means that your .class files have not been *preverified*. To run on CLDC VMs, Java class files must be processed by the preverifier tool in the Wireless Toolkit. Errors relating to "class format" or "class verification" may also indicate this problem.

A common error is to preverify, then obfuscate. Some obfuscators do not recognize CLDC stack maps, and remove them from the .class files to save space (effectively "un-preverifying" the classes). Preverify *after* you obfuscate. Some obfuscators (such as newer versions of Proguard) are able to preverify for you as part of the process.

Nothing Happens

Symptom: You launch the application, but nothing happens. No error, no exception, nothing.

If you have code in your MIDlet's constructor, try moving it to the startApp() method. Code in the constructor can sometimes behave oddly, since the MIDlet might not be created completely yet.

Avoid doing anything that takes a lot of time in startApp(). Like any event handler, startApp() should return as quickly as possible. Things that take a long time include:

- Loading many images

- Reading lots of records from RMS
- Reading files from the JAR
- HttpConnections

Create a separate Thread to perform these tasks. This also means you can provide some feedback to the user. Remember that calls to `Display.setCurrent()` are not acted upon immediately; you must return from `startApp()` before anything will be displayed.

No Luck?

Post a question in the Mobile Java Forum. Provide as much detail as possible, including any messages from the device, the device's model, and the contents of your JAD and MANIFEST files.