**IBM WebSphere Everyplace Micro Environment
Runtime for Windows Mobile 5.0 CLDC 1.1/MIDP 2.0**

IBM

WebSphere® software

Installation Guide

Version 6.1

Java
POWERED

Note: Before using this information and the product it supports, read the general information in the Notices section of this guide.

Contents

Preface

IBM® WebSphere® Everyplace® Micro Environment (WEME) is a Java™ 2 Micro Edition (J2ME™) certified "Java Powered" runtime environment that provides the foundation for the deployment of applications to a variety of mobile devices. This release optimizes the WEME runtime environment for Windows Mobile™ 5.0 operating systems.

This guide explains how to deploy this WEME release to Windows Mobile 5.0 platforms.

Who should read this guide

This guide is intended for application programmers developing embedded applications for this product.

Readers should be familiar with the following:

- Windows Mobile 5.0: ActiveSync® and File Explorer
- Java related concepts, terminology, and programming fundamentals

What this guide contains

This book contains the following sections:

- **Introducing WebSphere Everyplace Micro Environment 6.1** — Introduces WEME and the J9 Virtual Machine (VM). It also describes the release package contents and system requirements necessary for a successful product installation.

- **Deploying J9 to Windows Mobile 5.0** — Describes how to deploy J9 to Windows Mobile 5.0 target systems.

- **Running a demo MIDlet** — Describes how to run a sample application.

- **J9 runtime files** — Provides a list of files in the J9 runtime and describes what they are used for.

- **J9 command options** — Provides a list of J9 command line options.

Contacting software support

Before contacting IBM Software Support with a problem, refer to the IBM WEME Software Support site at the following Web site:
http://www.ibm.com/software/wireless/weme/support.html

On this Web site, you can search for technical notes, white papers and other content related to the IBM WEME. For additional help, contact software support by using the methods described in the *IBM Software Support Guide* at the following Web site:
http://techsupport.services.ibm.com/guides/handbook.html

The guide provides the following information:

- Registration and eligibility requirements for receiving support
- Telephone numbers, depending on the country in which you are located

Conventions used in this guide

The following typeface conventions are used in this guide:

Bold Lowercase commands or mixed case commands that are difficult to distinguish from surrounding text, keywords, parameters, options, names of Java classes, and objects are in bold.

Italic Variables, titles of publications, and special words or phrases that are emphasized are in italic.

`Monospace` Code examples, command lines, screen output, file and directory names that are difficult to distinguish from surrounding text, system messages, text that the user must type, and values for arguments or command options are in monospace.

Introducing WebSphere Everyplace Micro Environment 6.1

IBM WebSphere Everyplace Micro Environment (WEME) 6.1 provides the underlying platform for the deployment of e-business applications to small mobile devices.

The J9 VM is the core of WEME. It is the IBM implementation of the Java Virtual Machine (JVM) Specification, Version 1.4. For more on this Java VM Specification, see the following Web site: http://java.sun.com/docs/books/vmspec/

The J9 runtime environment consists of the J9 VM and Java Class Libraries (JCL). It is J2ME compliant and contains Connected Limited Device Configuration (CLDC) 1.1 and Mobile Information Device Profile (MIDP) 2.0 based technologies. The WEME product is supported on a variety of operating systems and hardware architectures. This document covers WEME 6.1 for Windows Mobile 5.0. For information on other platforms contact your IBM Sales Representative.

System requirements

This section lists the minimum product levels you should have installed.

Host system requirements:

- x86-architecture based system running Windows XP service pack 2

- ActiveSync 4.0

Target system requirements:

PDA or Smartphone device running Windows Mobile 5.0

Package contents

This release of WEME is available from the IBM Workplace Client Technology, Micro Edition Web site at: http://www.ibm.com/software/wireless/weme/

This package includes the following files:

Runtime files

- `\bin`
  Includes J9 programs and shared libraries

  Note: These files are listed and described in the **J9 runtime files** section of this document.

Classes and resources

- `\lib`
  Includes `charconv.zip`, `j2me.keystore,` and `security.policy`

- `\lib\jclMidp20`
  Includes `AMS.jad` and `jclMidp20.jxe`. The `AMS.jad` file is used by the system to launch the MIDP Application Management System (AMS). The AMS system allows a user to install, configure, launch, and delete MIDlets. The `jclMidp20.jxe` file contains all class libraries for CLDC 1.1 and MIDP 2.0 in the JXE format.

- `\lib\jclMidp20\ext`
  Includes `fontPaths.jar`. This JAR file contains a manifest and the `J9ext` file. The `J9ext` file is automatically read by the VM at runtime and it defines system properties which specify the default MIDP font file locations. For PDA devices, this file points to Tahoma and Tahomabd fonts. For Smartphone devices, the Nina and Ninab fonts are pointed to. If your system does not have these fonts installed, you may need to modify `J9ext` to point to different file locations.

- `\examples`
  Includes `GolfScoreTrackerSuite.jar` and `GolfScoreTrackerSuite.jad`. These files are needed to launch the GolfScoreTrackerSuite example.

- `\doc`
  Includes this platform installation guide

Java source

- `\lib`
  Includes `charconv-src.zip`

- `\lib\jclMidp20\source`
  Includes `source.zip`

- `\licenses`
  Includes licenses, notices and associated files

Deploying J9 to Windows Mobile 5.0

Use the following steps to deploy this WEME Runtime for Windows Mobile 5.0 CLDC 1.1/MIDP 2.0 on your target device.

1. Download the product Installer to the Host PC from the following Web site: http://www.ibm.com/software/wireless/weme/.

   Note: Three versions of the product installer are available. One version supports PDA devices with VGA/Hi-Res, another supports PDA devices with QVGA, and the last supports Smartphone devices.

   - PDA with VGA/Hi-Res support: `ibm-weme-wm50-arm-vga-midp20-6.1.0-YYYYMMDD-######-###.exe`

   - PDA with QVGA support: `ibm-weme-wm50-arm-midp20-6.1.0-YYYYMMDD-######-###.exe`

   - Smartphone support: `ibm-weme-wm50-sp-arm-midp20-6.1.0-YYYYMMDD-######-###.exe`

   Note: For Linux x86 host platforms, download the corresponding `.bin` file.

2. Use the **Extraction Wizard** to extract the files. Extract the files to the default location indicated.

3. Select the Explore icon in **ActiveSync** to open a **File Explorer** window for your mobile device.
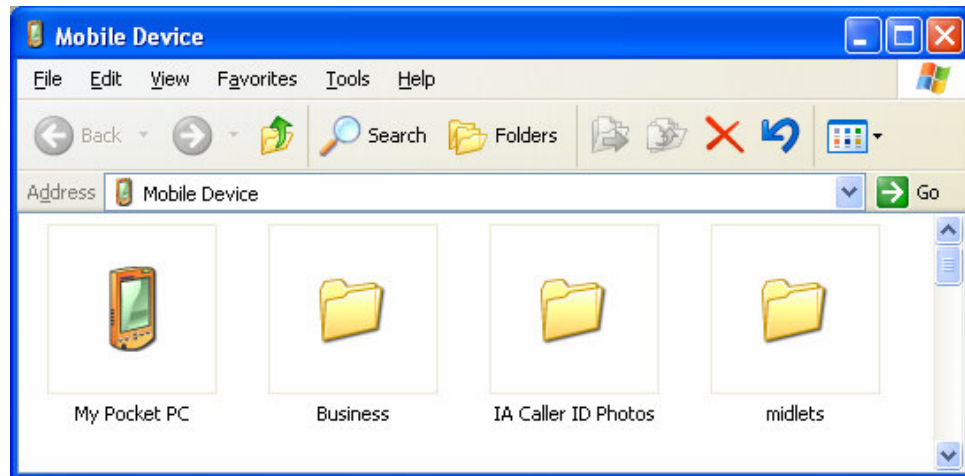
4. Use **File Explorer** to go to the `root` folder on your mobile device and create a new folder named `J9`.

5. Open the `J9` folder and create a `MIDP20` folder.

6. Copy the previously unzipped `bin`, `lib` and `examples` directories to the `MIDP20` folder.

   Note: These directories can be copied to any location you choose on the target file system; however, the `\bin` and `\lib` directories must remain intact. The base J9 executable and its shared objects must remain in the `\bin` directory.
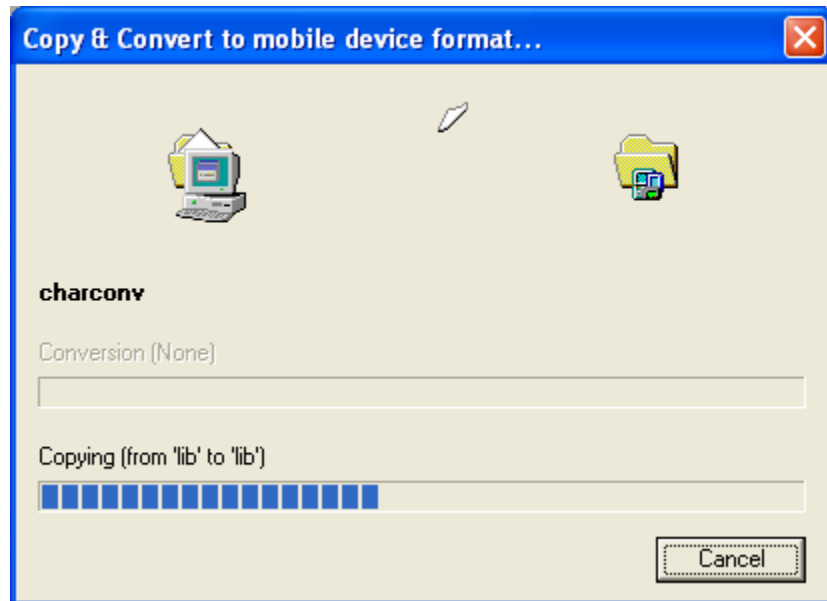
Installing and running a demo MIDlet

A demo MIDlet is included with this product. The JAD and JAR files needed to run this MIDlet are located in the following directory:

```
%JAVAHOME%\examples\GolfScoreTrackerSuite.jad
%JAVAHOME%\examples\GolfScoreTrackerSuite.jar
```
Note: `%JAVAHOME%` is the location of the J9 VM installation.

To install and run this demo MIDlet on the target file system:

1. In **File Explorer**, go to the `\J9\MIDP20\bin` folder and tap `emulator` to run this program. The **MIDlet List** window is now displayed.

2. Tap the **Install** button.

3. Select the keyboard function and type the following in the **URL** text field:
   `file:///J9/MIDP20/examples/GolfScoreTrackerSuite.jad`

4. Tap the **Ok** button. The **Progress** dialog displays this message:
   Installing a Midlet from
   `"file:///J9/MIDP20/examples/GolfScoreTrackerSuite.jad"`

5. Tap the **Yes** button. Ignore the subsequent warning message and tap the **Yes** button. A **Success** message is now displayed.

6. Tap the **Ok** button. The **MIDlet List** window is displayed. To run the MIDlet, tap `GolfScoreTrackerSuite` and tap the **Launch** button. An application window similar to the following appears.
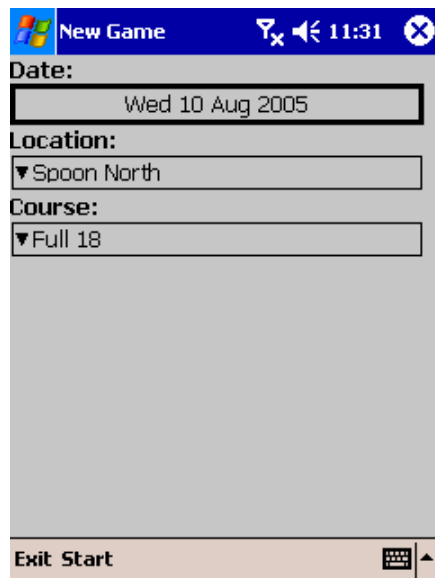
J9 runtime files

The J9 runtime is a feature-rich VM that you can customize to fit the needs of a specific application. Depending on which runtime is used, all of the files may or may not be in the directory structure. Before loading the runtime files onto the target, you can remove some files from the runtime image to minimize its size. The \bin directory includes a variety of programs and shared libraries. This section organizes the J9 runtime files into two categories:

- **Files required on a target**

- **Optional files on a target**

Files required on a target

This section specifies the absolute minimum set of files that must be available on a running target.

ivemidp20_23.dll	Contains the AMS natives, the Limited Connected Device User Interface (LCDUI) natives, and possible media natives depending on the platform.
j9vmall23.dll	All Core DLL files combined into one DLL file for platforms with a limited number of open shared libraries, like the Pocket PC. It includes:

iverel23.dll	Specifies JXE support, including the relocator and JXE files class loader natives.
j9bvc23.dll	Enables the byte code verifier required when J9 is started with the (default) **–verify** option. Use **–noverify** when removing this file.
j9dyn23.dll	Specifies the dynamic class loader for class files loaded from directories, JAR files or JXE files.
j9gc23.dll	Specifies the Garbage Collector.
j9hookable23.dll	Specifies the link library for hookable components, such as the garbage collector event handling, Just in Time (JIT) compiler, Ahead of Time (AOT) runtime, the debugger, Java Virtual Machine Profiler Interface (JVMPI), MicroAnalyzer, SmartLinker profiler, and verbose output.

	`j9prt23.dll`	Specifies the J9 port library, containing target operating system dependent code.
	`j9thr23.dll`	Defines support for implementing Java threads in either native OS threads or green threads.
	`j9vm23.dll`	Specifies the J9 VM, including implementations of the Java byte codes.
	`j9zlib23.dll`	Specifies the zlib data compression library (`(C) 1995-2002 Jean-loup Gailly and Mark Adler`). This file is required when using compressed JAR files.
`jclmidp20_23.dll`		Natives for the JCL. The JCL natives are provided in one file specific to the JCL (CLDC 1.1/MIDP 2.0) used.

Note: At least one of the following listed launcher programs is required.

`emulator.exe`	Universal Emulator Interface program that allows the user to control the Application Management System (AMS) through a standard specification. (MIDP only)
`j9.exe`	The J9 VM program.
`j9midp20.exe`	Starts the J9 VM with specified options required for certification of the J2ME MIDP class library.
`j9w.exe`	J9 VM program. Opens J9 VM without a console.

Optional files on a target

This section specifies files that are needed only when you want to enable related J9 options or features.

`j9dbg23.dll`	Defines support for debugging the target. This is required when J9 is started with the **–debug** option. This file requires either `j9dbgi23.dll` or `j9rdbi23.dll`.
`j9mjit23.dll`	Used if **-Xmjit** option is specified.
`j9mjitd23.dll`	Problem Determination Library for MicroJIT.
`j9rdbi23.dll`	Specifies access to the remote debug server, which is required when J9 is started with the following options: **–debug**:*options* **–Xrdbginfo**:*host*:*port*
`java.properties`	Contains all externalized text messages, such as help text and error messages for the default language (English). If this file is missing, numeric error codes are printed.

J9 command options

This section discusses command line options used with **emulator.exe**. It also discusses the common and advanced J9 v2.3 command line options used with **j9.exe**, **j9w.exe** and **j9midp20.exe**.

Note: Both the Pocket PC and Smartphone platforms require that you use fully qualified path names. Relative path names will not work.

Emulator command options

This section describes command line options used with **emulator** for J9 v2.3.

Syntax:

emulator [*AMS VM options...*] [*MIDlet VM options...*] [*AMS option*]

The following are used to specify VM Options:

–J*AMS_VM_Arg*	Specifies a VM option for the VM running the AMS MIDlet. For example: `-J-Xint` would run the AMS VM in interpreted mode. Any standard J9 option can be passed in this way.
–Xj9arg:*MIDlet_VM_Arg*	MIDlets launched from the AMS run in a new VM. This option specifies VM options for the launched VM. For example: `-Xj9arg:-Xint` would run the new VM in interpreted mode. Any standard J9 option can be passed in this way.

The following are used to specify AMS Options:

–classpath *path*	Specifies the classpath used to launch a MIDlet. When this option is specified, it must be followed by a MIDlet class name. For example: **emulator –classpath** *path MidletClassName*
–help	Print a message explaining the available options.
–D*property*	Sets a system property for the VM running the AMS.
–version	Prints out the emulator version string.
–Xdebug	Enables runtime debugging. If this option is specified, **–Xrunjdwp** must also be specified.

–Xdescriptor:*jadURL*	Launches a transient MIDlet. The AMS will install the MIDlet at the URL given and then launch it. When the MIDlet is closed, it will be uninstalled and any data associated with the MIDlet will be deleted. The *jadURL* argument must be a fully formed URL.
–Xjam*: command[=application]*	Passes an argument to the Application Manager. If no command is specified, this simply brings up the Application Manager UI. The commands are as follows:

install=*jadURL*	The AMS installs the MIDlet at the URL specified by *jadURL*.
force=*value*	Specifies whether the AMS should overwrite an existing MIDlet when installing a MIDlet. *value* should be either "true" or "false".
list	Prints a list of the installed MIDlets.
storagenames	Prints the storage names of the installed MIDlets. These are unique identifiers assigned to MIDlets during the installation process.
run=*application*	Run an installed application. The *application* argument must be a valid storage name or a valid MIDlet index.
remove=*application*	Remove an installed application. The *application* argument must be a valid storage name or a valid MIDlet index.
transient=*jadURL*	Install and run the MIDlet specified by the *jadURL* argument. The MIDlet and its data are removed when the MIDlet is closed.

–Xrunjdwp:*name=value*	Sets properties for runtime debugging. The options are as follows:

address=*host:port*	Specifies the address of the debug server to connect to.
server=*value*	Specifies whether to run as a debug server or not. The *value* must be "y" or "n". If "y" is specified, the application will wait for a debugger connection before running the MIDlet. The default value is "n".
suspend=*value*	Specifies whether to suspend the VM as soon as a debugger connection is made. The *value* argument must be "y" or "n". The default value is "y".

Common options used with j9.exe, j9w.exe and j9midp20.exe

This section describes common J9 v2.3 command line options used with `j9.exe`, `j9w.exe` and `j9midp20.exe`.

Syntax:

j9 -jcl:midp20 [*options…*] **–classpath** *path_to_MIDlet_jar* **–jxe**:*path_to_jclMidp20.jxe_file path_to_MIDlet_jad* [*midlet arguments…*]

> **Note:** If you use the **–Xbootclasspath** option instead of the **–jxe:** option, you must specify the path to the `jclMidp20.jxe` file and the `AppManager` startup class. For example:
> **j9 -jcl:midp20** [*options…*] **–classpath** *path_to_MIDlet_jar*
> **–Xbootclasspath:**`%JAVAHOME%\lib\`
> `jclMidp20\jclMidp20.jxe javax.microedition.lcdui.AppManager`
> *path_to_MIDlet_jad* [*MIDlet arguments…*]

j9w -jcl:midp20 [*options…*] **–classpath** *path_to_MIDlet_jar* **–jxe**:*path_to_jclMidp20.jxe_file path_to_MIDlet_jad* [*midlet arguments…*]

j9midp20 [*options…*] **–classpath** *path_to_MIDlet_jar* **–jxe**:*path_to_jclMidp20.jxe_file path_to_MIDlet_jad* [*midlet arguments…*]

–? or **–help**	Displays help for J9 standard command options.
–classpath *path* **–cp** *path*	Either **–classpath** *path* or **–cp** *path* can be used to set a class path for this invocation of J9. The final value of **–classpath** is determined as follows:

- If the **–classpath** option is set, its value is used.

- If the **–classpath** option is not set, and the CLASSPATH environment variable is set, then its value is used.

- If neither of the preceding are set, the current directory (`.`) is used.

If the class path includes:

- Multiple class path entries, separate them with a semicolon ";"

- A JAR, ZIP or JXE file, add the full name of the file to the class path

- CLASS files, specify the top-level directory of the CLASS file tree

Example: `–classpath \myclasses;\myjars\foo.jar`
CAUTION: The J9 class libraries and the J9 VM are not compatible with other vendor's class libraries. Because it is

possible to have more than one runtime environment installed on your host computer, make sure that you do not mismatch these libraries when specifying the class path. In particular, if your CLASSPATH environment variable is set, ensure that other vendor's libraries are not on it.
Note: Java and javax class packages must be on **–Xbootclasspath**, not **–classpath**.

–jxe:*jxe_file*	Reads the specified JXE file, searching for the classes in this file. All classes found in the JXE file are placed at the end of the boot path. For example: `-jxe:hello.jxe` **Note:** When using the **–jxe:** option, do not specify the startup class. Specify the **–jxe:** option as the last option on the command line. **Note**: It is recommended that you use **–classpath** (where applicable) or **–Xbootclasspath** if the JXE file contains boot classes. See the **–Xbootclasspath** option for details.
–D*prop*=*value*	Sets the value of a system property. Example: **–**`Dmy.property=some.value` Sets the value of `my.property` to `some.value`. If no value is given, **–D***prop* sets the value to `null`. To set values for multiple system properties, repeat the option statement, using a space to separate statements. Example: `j9` **–**`Dprop1=val1` **–**`Dprop2=val2` **–**`Dprop3=val3` **Note:** Spacing is important in this option's syntax. There is no space between the initial **–D**, its property argument, the equal sign, or the *value* argument. Example: `-Dname=John_Smith` **Note:** If *value* contains spaces, enclose the option in double quotes. Example: `"-Dmy.property=value with space"`
–debug:*options*	Enables debug, Java Debug Wire Protocol (JDWP) standard *options*.
–jcl:*config*	This command option is used to specify which JCL shared library will provide JNI natives for the class library Java code. To use the MIDP 2.0 class libraries you must specify `-jcl:midp20`. The `-jcl:midp20` argument is only needed with using **j9**. When running **j9midp20,** the `-jcl:midp20` option is already specified by default. If the **–jcl** option is used without indicating a

	–Xbootclasspath:*path*, the value for *path* is assumed to be `%JAVAHOME%\lib\jclMidp20\jclMidp20.jxe`. However, if the class libraries are stored in a non-default location, you must include the **–Xbootclasspath:***path* option to direct the VM to the jclMidp20.jxe file. See the **–Xbootclasspath** option for details. **Note:** If the **–Xbootclasspath** and the **–jcl** VM options are mismatched, the VM generates an `Incompatible class library` error.
–verify	Enables class file (byte code) verification. **Note:** The **–verify** option is true by default. To disable byte code verification, specify **–noverify**.

–version	Each VM build is identified by a version string of the form:
	`YYYYMMDD_#####_flags`
	Example: `20051103_03795_lEdCmV`
	The first 8 digits indicate the date the VM was built on. The next 5 digits indicate the build ID.
	
	The flags indicate the configuration:
	    1st letter:
	        l: little endian
	        b: big endian
	        L: 64-bit little endian
	        B: 64-bit big endian
	    2nd letter:
	        E: emulated FPU
	        H: hardware FPU
	    3rd letter:
	        s: static linkage
	        d: dynamic linkage
	    4th letter:
	        C: CLDC
	        F: Foundation
	        S: J2SE
	    5th letter:
	        M: Desktop GC
	        m: Tiny GC
	        G: Embedded
	    6th letter:
	        i: no JIT
	        a: AOT only
	        r: large JIT
	        q: small JIT
	        V: MicroJIT
	        R: large JIT + MicroJIT
	        Q: small JIT + MicroJIT
	        A: AOT + MicroJIT

–X	Prints help for non-standard (advanced) options.

Advanced options used with j9.exe, j9w.exe and j9midp20.exe

This section describes advanced J9 v2.3 command line options used with `j9.exe`, `j9w.exe` and `j9midp20.exe`. These command line options are non-standard and subject to change without notice.

–Xbootclasspath:*path*	Sets the bootstrap class path to *path*. For example:
	`-jcl:midp20 -Xbootclasspath:%JAVAHOME%\lib\`
	`jclMidp20\jclMidp20.jxe`
	Note: When using this command line option, the **–jcl:LibraryName** option must also be used, as shown in the above example, to indicate which class library natives the application should use.
	Note: If you use the **–Xbootclasspath** option instead of the **–jxe:** option, you must specify the path to the JXE file and the startup class. For example:
	`j9 -jcl:midp20 -classpath myapps\MyMIDlet.jar`
	`-Xbootclasspath:%JAVAHOME%\lib\jclMidp20\`
	`jclMidp20.jxe javax.microedition.lcdui.`
	`AppManager \myapps\MyMIDlet.jad`

–Xbootclasspath/p:*path*	Prepends the classes in *path* to the bootstrap class path. This option is useful for applying temporary fixes and/or adding to the bootstrap class path.
–Xbootclasspath/a:*path*	Appends the classes in *path* to the bootstrap class path. This option is useful for applying temporary fixes to application classes and/or adding to the bootstrap classpath.
–Xdbg:*options*	Enables standard Java Debug Wire Protocol (JDWP) debug options.
–Xdbginfo:*symbol_file_path*	Enables the debug info server.
–Xrdbginfo:*host*:*port*	Enables the remote debug info server.
–Xrunjdwp:*options*	Enables standard JDWP debug options. **Note:** Starts a JDWP server.
–Xfuture	Turns on strict class-file format checks. These checks enforce closer conformance to the class-file format specification.
–Xss_x_	Sets the maximum Java thread stack size to *x*.
–Xmso_x_	Sets the operating system thread stack size to *x*.
–Xint	Run interpreted only.

–Xmjit[:**count**=*x*, **code**=*x*]

With no parameters, **–Xmjit** enables the MicroJIT. Useful parameters are:

count=*x*, where *x* is the upper limit of the number of times a method is invoked before it is compiled. **Example:** `-Xmjit:count=0,` forces the MicroJIT to compile everything on first execution.

code=*x*, where *x* sets the size of the MicroJIT code cache, in kilobytes. **Example:** `-Xmjit:code=1024,` sets the size of the MicroJIT code cache to 1MB. The code cache will not grow beyond 1MB.

The following MicroJIT command options specify if and when the code cache is to be flushed:

- `flushCodeCacheWhenFull`

- `flushCodeCacheOnPhaseShift`

- `noFlushCodeCache`

Flushing the code cache reverts all the compiled methods, including methods currently executing, to their initial interpreted state. Methods have to re-qualify for compilation. This allows the use of a smaller code cache, while preserving reasonable performance.

If **flushCodeCacheWhenFull** is in effect, the code cache is flushed when there is no more room in the code cache for a newly-compiled method.

If **flushCodeCacheOnPhaseShift** is in effect, the MicroJIT compiler tries to identify when there is a major change in the behaviour of the Java program, by tracking class load events. If the code cache is more than half full at such a point the code cache is flushed.

If **noFlushCodeCache** is in effect, the code cache is never flushed; when the code cache is full the methods already compiled remain in the code cache but no more methods are compiled.

flushCodeCacheWhenFull and **flushCodeCacheOnPhaseShift** are set by default on ARM, all operating systems, and x86 Windows. On other platforms, the default is **noFlushCodeCache**. Therefore, to set only one of those options on ARM or x86 Windows, the option should be specified with **noFlushCodeCache** preceding it. **Example:** `-Xmjit:noFlushCodeCache,` `flushCodeCacheWhenFull`

–Xoptionsfile=*filename*
StartupClass [arguments..]

J9 VM Version 2.3 on all platforms supports an option file for the purpose of reducing the length of the command line.

Example: `%JAVAHOME%\bin\j9 –Xoptionsfile=`
`vm.options com.ibm.myapps.MyApp1`

- An option file is a text file with one option per line.

- Lines starting with # are ignored and can be used for comments.

- The \ character can be used as a continuation so that a single option can span multiple lines.

- The following command line options must be converted into their J9 internal form when used in the options file:

Command line option	J9 internal form
–analyze	–Xanalyze:NULL
–analyze:	–Xanalyze:
–classpath *path*	–Djava.class.path=*path*
–dbginfo:	–Xdbginfo:
–debug:	–Xdbg:
–jcl:*config*	–Xjcl:jcl*config*_23
–noverify	–Xverify:none
–rdbginfo:	–Xrdbginfo:
–verify	–Xverify
–verify:	–Xverify:
–Xrunjdwp:	–Xdbg:

- The following options must be entered on the command line (they are ignored when listed in the options file):

     –jar
     –jxe
     –jxe:
     –jxespace:
     –Xoptionsfile=
     Note: Embedded options files are not supported.

- Undocumented options are ignored when listed in the options file. To be used, they must be added to the command line.

- All other options, such as **–D**, **–Xint**, **–Xmx** etc. are the same on the command line and in the options file.

Environment variables are not supported in the options file. For example the following works on the command line but not in the options file : **–**`Dmy.property=some.value`

- Options listed in the option file override options on the command line regardless of their position.

Example: `vm.options` file:

```
-Xint
-Xanalyze:st=true,ia=192.168.1.100,ms=100000
-Djava.security.manager
-Djava.security.policy=my.policy
#-Djava.class.path=my.jar
```

–Xrun*dll*[:*options*]   Loads helper libraries, such as those used with JVMPI.

J9 V2.3 GC command line options

The Garbage Collector for IBM WEME Runtime for Windows Mobile 5.0 CLDC 1.1/MIDP 2.0 provides global GC with compaction. The following related command line options are provided:

Option parameter key

x	integer value in bytes, or append with 'k' or 'M' for large values
percentage	integer value in the range of 0--100 (inclusive)
age	integer value in the range of 1--14 (inclusive)
time	integer value (in milliseconds)

–Xmx*x*	Sets memory object heap memory size to *x*. **Xmx** >= **NewSpace** size plus **OldSpace** size Scavenger enabled: minimum size 1536 bytes on 32-bit architectures, 6072 bytes on 64-bit architectures Scavenger disabled: minimum size 512 bytes on 32-bit architectures, 2048 bytes on 64-bit architectures
–Xms*x*	Sets the initial memory size to *x*. scavenger enabled: minimum size 4 kilobytes on 32-bit architectures, 8 kilobytes on 64-bit architectures **Xms** >= **Xmn** + **Xmo** scavenger disabled: minimum size 4 kilobytes on 32-bit architectures, 8 kilobytes on 64-bit architectures **Xms** >= **Xmos**
–Xmos*x*	Sets the initial **OldSpace** size to *x*. **Note:** Minimum size 512 bytes on 32-bit architectures, 2048 bytes on 64-bit architectures
–Xmox*x*	Sets the maximum **OldSpace** size to *x*.
–Xmo*x*	Sets the initial and maximum **OldSpace** size to *x*. **Note:** Attempts to set **–Xmo** and **–Xmos**, or **–Xmo** and **–Xmox** are rejected

–Xmca$x$	Sets the RAM class segment increment to x.
	RAM class segments contain the portion of the Java classes that needs to be modified at runtime, such as the pointers to the class loader, super classes, implemented interfaces, statics, first instance and so forth. There is at least one RAM class segment per Class Loader. If more space for the RAM classes is needed, the J9 VM allocates a new segment in the same size.
–Xmco$x$	Sets the ROM class segment increment to x.
	The code of Java classes loaded from class files is stored in special ROM classes segment types called Dynamically Loaded Classes, whose size is determined by this parameter. There is at least one ROM class segment per class loader. If more space for the dynamically loaded classes is needed, the J9 VM allocates a new segment in the size determined by the class loader.
	Note: This option does not apply to classes loaded from a JXE file. The ″rom.classes″ entry in the JXE file is mapped directly to a ROM class segment.
–Xmoi$x$	Sets the **OldSpace** increment to x. This value is used to expand the **OldSpace.** A value of 0 means no expansion is allowed. If **–Xmoi** is not specified, there are no restrictions on the expansion size of **OldSpace**.

Notices

This information was developed for products and services offered in the U.S.A. IBM might not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM might have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms.

These examples have not been thoroughly tested under all conditions.

No warranty

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CAN NOT BE EXCLUDED, IBM MAKES NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THE WARRANTY OF NON-INFRINGEMENT AND THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY. IBM MAKES NO WARRANTY REGARDING THE CAPABILITY OF THE PROGRAM TO CORRECTLY PROCESS, PROVIDE AND/OR RECEIVE DATE DATA WITHIN AND BETWEEN THE 20TH AND 21ST CENTURIES.

The exclusion also applies to any of IBM's subcontractors, suppliers, or program developers (collectively called "Suppliers").

Limitation of Liability

NEITHER IBM NOR ITS SUPPLIERS WILL BE LIABLE FOR ANY DIRECT OR INDIRECT DAMAGES, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST SAVINGS, OR ANY INCIDENTAL, SPECIAL, OR OTHER ECONOMIC CONSEQUENTIAL DAMAGES, EVEN IF IBM IS INFORMED OF THEIR POSSIBILITY. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE EXCLUSION OR LIMITATION MAY NOT APPLY TO YOU.

You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
IBM logo
Everyplace
WebSphere

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. The Java Powered logo is used under license from Sun Microsystems, Inc.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Notices for Excluded Components

These notices are provided in the `notices.txt` file, located in the `\licenses` directory.

IBM ®