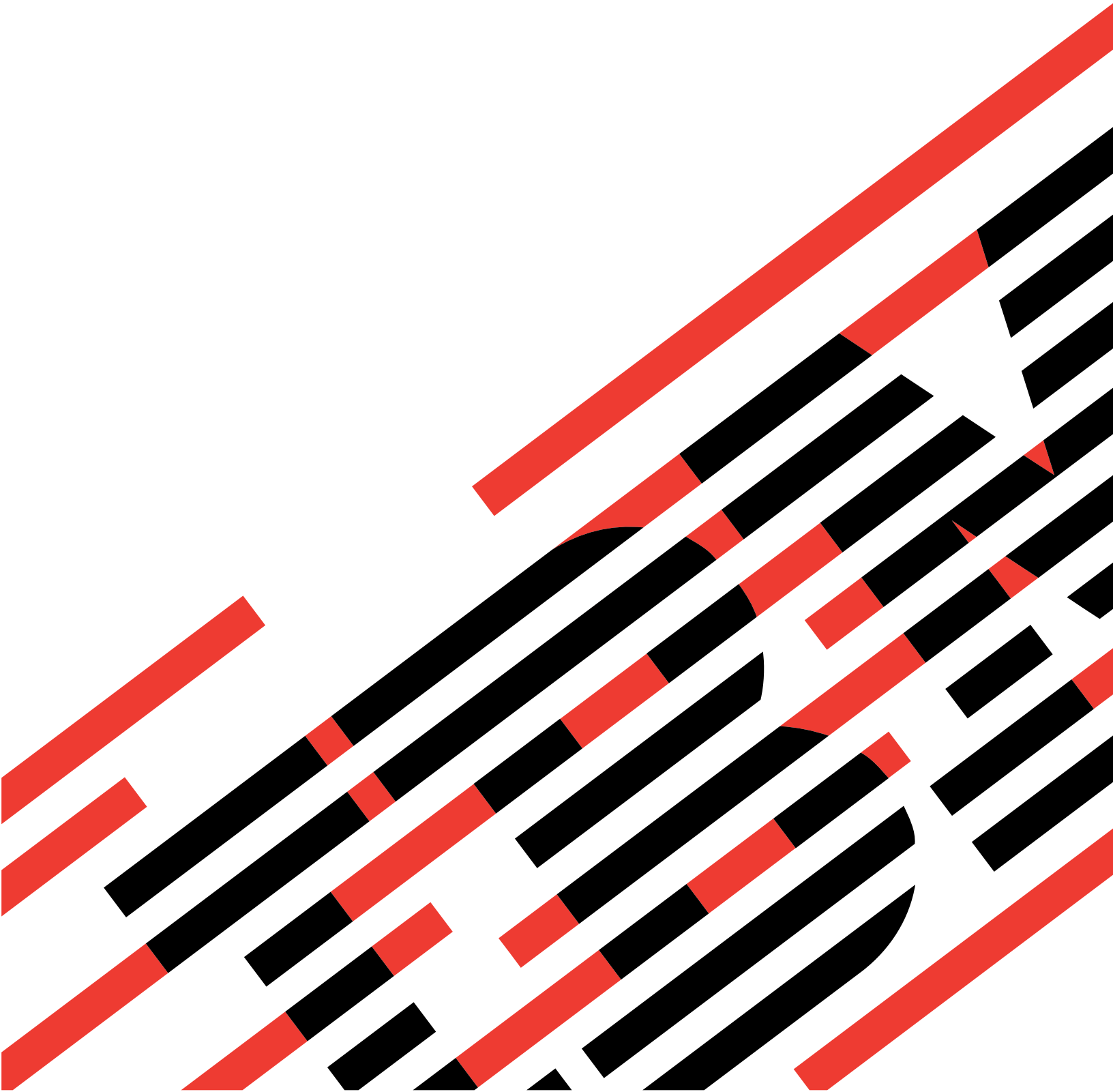
IBM

eserver

iSeries

Developer Kit for Java Commands

Version 5 Release 3



IBM

@server

iSeries

Developer Kit for Java Commands

Version 5 Release 3

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 17.

First Edition (May 2004)

This edition applies to version 5, release 3, modification 0 of Developer Kit for Java (product number 5722-JV1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CICS models.

Contents

Run Java Program (JAVA)

Where allowed to run: All environments (*ALL)
Threadsafe: No

Parameters
Examples
Error messages

The Run Java Program (JAVA) command runs the Java program associated with the specified Java class. If no Java program exists, one is created and associated with the class file.

This command can operate on files in any file system that supports the integrated file system APIs.

Top

Parameters

Keyword	Description	Choices	Notes
CLASS	Class file or JAR file	*Path name*, *VERSION	Required, Positional 1
PARM	Parameters	Single values: ***NONE** Other values (up to 200 repetitions): *Character value*	Optional, Positional 2
CLASSPATH	Classpath	*Path name*, ***ENVVAR**	Optional
CHKPATH	Classpath security check level	***WARN**, *SECURE, *IGNORE	Optional
OPTIMIZE	Optimization	***JIT**, *INTERPRET, 10, 20, 30, 40	Optional
INTERPRET	Interpret	***OPTIMIZE**, *YES, *NO, *JIT	Optional
PROP	Properties	Single values: ***NONE** Other values (up to 100 repetitions): *Element list*	Optional
	Element 1: Property name	*Path name*	
	Element 2: Property value	*Character value*, ***NONE**	
GCHINL	Garbage collect initial size	256-240000000, ***DFT**	Optional
GCHMAX	Garbage collect maximum size	256-240000000, ***DFT**, *NOMAX	Optional
GCFRQ	Garbage collection frequency	0-100, **50**	Optional
GCPTY	Garbage collection priority	**20**, 10, 30	Optional
OPTION	Option	Values (up to 4 repetitions): ***NONE**, *VERBOSE, *DEBUG, *VERBOSEGC, *NOCLASSGC	Optional
JOB	Job name	*Name*, **QJVACMDSRV**, *GEN	Optional
AGTPGM	Agent program	Single values: ***NONE** Other values: *Qualified object name*	Optional
	Qualifier 1: Agent program	*Name*	
	Qualifier 2: Library	*Name*, ***LIBL**, *CURLIB	
AGTOPTIONS	Agent options	*Character value*, ***NONE**	Optional
OUTPUT	Output	Single values: *PRINT, *NONE Other values: *Element list*	Optional
	Element 1: Destination	*	
	Element 2: Program end action	***PAUSE**, *CONTINUE	

Top

Class file or JAR file (CLASS)

Specifies the class name or jar file to be run. The class name may be qualified by one or more package names. Each package name must be followed by a period. For example, CLASS('pkg1.pkg2.myclass') identifies a class qualified by two package names.

A jar file name may be specified only when running JDK 1.2 or higher. The start up class must be indicated by the Main-Class in the manifest header.

class-name
Specify the name of the class to be run.

jar-name
Specify the name of the jar file with the Main-Class specified in the manifest.

***VERSION**
The build version information for the Java Development Kit (JDK) and the Java Virtual Machine (JVM) is displayed. No Java program is run.

Top

Parameters (PARM)

Specifies one or more parameter values that are passed to the Java program. A maximum of 200 parameter values can be passed.

***NONE**
There are no input parameters to the Java program.

parameter-value
Specify the parameter value to be passed to the Java program.

Top

Classpath (CLASSPATH)

Specifies the path used to locate classes. Directories are separated by colons.

***ENVVAR**
The class path is determined by the environment variable CLASSPATH.

class-path
Path used to locate classes. An example class path is '/directory1/directory2:/QIBM/ProdData/Java400'.

Top

Classpath security check level (CHKPATH)

Specifies the level of warnings given for directories in the CLASSPATH that have public write authority. A directory in the CLASSPATH that has public write authority is a security exposure because it may contain a class file with the same name as the one you want to run. Whichever class file is found first is run.

***WARN**
A warning message is sent for each directory in the CLASSPATH that has public write authority.

***SECURE**
A warning message is sent for each directory in the CLASSPATH that has public write authority. If one or more warning messages are sent, an escape message is sent and the Java program is not run.

***IGNORE**
Ignore the fact that directories in the CLASSPATH may have public write authority. No warnings messages are sent.

Top

Optimization (OPTIMIZE)

Specifies how to treat class files if no Java program is associated with the file.

For 10,20,30,40 this specifies the optimization level of the Java program which will be created if no Java program is associated with the Java class file. The Java program will contain machine instruction sequences that are run when the Java program is invoked and will remain associated with the class file after the Java program has been run.

If the Java class file is determined to be downlevel or out of date, a new Java program will be created using the optimization level that was used when the previous Java program was created, instead of the value specified for this parameter.

For OPTIMIZE(*INTERPRET), the resulting Java program interprets the class byte codes when invoked.

OPTIMIZE(*INTERPRET) Java programs will be smaller but will run slower than Java programs created with higher optimization levels. As you increase the optimization level beyond 10, the Java program performance will generally improve, but the time required to create the Java program will increase and you will have less ability to debug the Java program.

For OPTIMIZE(*JIT), no Java program containing machine instruction sequences is created if no program is associated with the class file. Rather, the class is run using the Just In Time compiler (JIT).

***JIT** No Java program containing machine instruction sequences is created. The class is run using the Just In Time compiler (JIT).

***INTERPRET**
The Java program created does not contain machine specific instructions. It will be interpreted when the program is started. Variables can be displayed and modified while debugging.

If *OPTIMIZE is specified for the **Interpret (INTERPRET)** parameter, all of the classes that run will be run interpreted even if there is an optimized Java program associated with the class.

10 The Java program contains a compiled version of the class file byte codes and has only minimal additional compiler optimization. Variables can be displayed and modified while debugging.

20 The Java program contains a compiled version of the class file byte codes and has some additional compiler optimization. Variables can be displayed but not modified while debugging.

30 The Java program contains a compiled version of the class file byte codes and has more compiler optimization than optimization level 20. During a debug session, user variables cannot be changed, but can be displayed. The presented values may not be the current values of the variables.

40 The Java program contains a compiled version of the class file byte codes and has more compiler optimization than optimization level 30. All call and instruction tracing is disabled.

Top

Interpret (INTERPRET)

Specifies how the Java class files should be run.

***OPTIMIZE**
Whether all Java classes are run interpretively depends on the value specified for the OPTIMIZE parameter. If OPTIMIZE(*INTERPRET) was specified, all Java classes will be run interpretively. If any other value was specified for the OPTIMIZE parameter, only Java classes with Java programs created using CRTJVAPGM command and specifying OPTIMIZE(*INTERPRET) will be run interpretively.

***YES** All Java classes will be run interpretively regardless of the OPTIMIZE value associated Java program. Java classes that need a Java program created will use the optimization level specified in the OPTIMIZE parameter.

***NO** Only Java classes with Java programs created using CRTJVAPGM command and specifying OPTIMIZE(*INTERPRET) will be run interpretively. Java classes that need a Java program created will be created with the optimization level specified in the OPTIMIZE parameter.

***JIT** All Java class files will be run using the Just In Time compiler (JIT), regardless of what OPTIMIZE value was used when the associated Java program was created.

Top

Properties (PROP)

Specifies a list of values to assign to Java properties. Up to 100 properties can have a value assigned.

Single values

***NONE**
No properties are specified.

Element 1: Property name

property-name
Specify the name of the property to be changed.

Element 2: Property value

property-value
Specify the value to be assigned to the property.

Top

Garbage collect initial size (GCHINL)

Specifies the initial size, in kilobytes, of the garbage collection heap. This is used to prevent garbage collection from starting on small programs.

***DFT** The default initial size is 2048 kilobytes unless it is overridden by a property.

256-240000000
Specify the initial size, in kilobytes, of the garbage collection heap. It is recommended that the initial heap size be set to 2048 kilobytes (the default) or larger.

Top

Garbage collect maximum size (GCHMAX)

Specifies the maximum size, in kilobytes, that the garbage collection heap can grow to. This is used to prevent runaway programs that consume all of the available storage. Normally, garbage collection runs as an asynchronous thread in parallel with other threads. If the maximum size is reached, all other threads are stopped while garbage collection takes place. This may adversely affect performance.

***DFT** The default for the parameter is used. The default maximum is determined by the system unless a property is specified. The heap will grow until all system resources are depleted. Then a synchronous garbage collection is started to reclaim resources no longer in use.

***NOMAX**
The maximum size is not specified by the user. The maximum is determined by the system. The heap will grow until all system resources are depleted. Then a synchronous garbage collection is started to reclaim resources no longer in use.

256-240000000
Specify the size, in kilobytes, that the garbage collection heap can grow to.

Top

Garbage collection frequency (GCFRQ)

Specifies the relative frequency that garbage collection runs.

This parameter is no longer supported. It exists solely for compatibility with the releases earlier than Version 4 Release 3 Modification 0 of the server.

Top

Garbage collection priority (GCPTY)

Specifies the priority of the tasks running garbage collection.

This parameter is no longer supported. It exists solely for compatibility with the releases earlier than Version 4 Release 3 Modification 0 of the server.

Top

Option (OPTION)

Specifies special options used when running the Java class.

***NONE**
No special options are used when running the Java class.

***VERBOSE**
A message is displayed each time a class file is loaded.

***DEBUG**
Allows the system debugger to be used for this Java program. This value cannot be used with the JIT compiler. That is, *DEBUG is mutually exclusive with INTERPRET(*JIT). Also, *DEBUG cannot be used when a JAR file is specified for the CLASS keyword.

***VERBOSEGC**
A message is displayed for each garbage collection sweep.

***NOCLASSGC**
Unused classes are not reclaimed when garbage collection is run.

Top

Job name (JOB)

Specifies the name that is associated with the batch immediate (BCI) job that is started when this command is run. The BCI job is where the Java program will be run.

QJVACMDSRV
The job name for the BCI job will be QJVACMDSRV.

***GEN** The job name is generated from the class name.

name Specify the name to be used for the BCI job that is used to run the Java program.

Top

Agent program (AGTPGM)

Specifies an ILE service program or OS/400 PASE program that contains a virtual machine (VM) agent. For example, a VM agent can be a remote debugger or profiler. The VM loads the agent program and looks for the entry point:

```
jint JNICALL JVM_OnLoad(JavaVM *jvm, char *options, void *reserved);
```

The VM calls the JVM_OnLoad function, passing a pointer to the JavaVM instance as the first argument, and the character string specified in the AGTOPTIONS parameter as the second argument. The third argument to JVM_OnLoad is reserved and set to NULL.

Single values

***NONE**
No VM agent program is specified.

Qualifier 1: Agent program

name Specify the name of the VM agent program.

Qualifier 2: Library

***LIBL** All libraries in the library list for the current thread are searched until the first match is found. If the VM agent is an OS/400 PASE program, the directories contained in the LIBPATH and PASE_LIBPATH environment variables are used to locate the program.

***CURLIB**
The current library for the thread is searched. If no library is specified as the current library for the thread, the QGPL library is searched.

name Specify the name of the library to be searched.

Top

Agent options (AGTOPTIONS)

Specifies a character string of the virtual machine (VM) agent options. A pointer to this character string will be passed as the second argument to the JVM_OnLoad function in the VM agent program specified in the AGTPGM parameter. The format of the options string is determined by the agent program.

***NONE**
No options are specified. A NULL options argument will be passed to the JVM_OnLoad function.

character-value
Specify the options string to be passed to the VM agent program on start-up.

Top

Output (OUTPUT)

Specifies where output from the Java program should be sent and, if output is directed to the Java shell display, whether the shell display panel should go away when the Java program ends.

Single values

***PRINT**
The Java program output is sent to a spooled file through the QPRINT printer device file.

***NONE**
The Java program output is discarded.

Element 1: Destination

*****
A Java shell display panel will be used to display output if the Java program is run from an interactive job. If the Java program is run in a batch job, the Java program output is sent to a spooled file through the QPRINT printer device file.

Element 2: Program end action

***PAUSE**
The Java shell display panel is shown until the F3, F12, or Enter key is pressed.

***CONTINUE**
The Java shell display panel is closed (goes away) when the Java program ends.

Top

Examples

Example 1: Run a Java Program

```
JAVA   CLASS('projectA.myJavaclassname')
```

This command will run the iSeries Java program associated with the class *myJavaclassname*. The job name of the batch immediate (BCI) job where the Java program will run will be QJVACMDSRV.

Example 2: Generate the Job Name for the Java Program

```
JAVA   CLASS('projectA.myJavaclassname') JOB(*GEN)
```

This command will run the iSeries Java program associated with the class *myJavaclassname*. The job name of the batch immediate (BCI) job where the Java program will run will be MYJAVACLAS.

Top

Error messages

***ESCAPE Messages**

JVAB534
Unable to complete Java program ″&1″.

JVAB535
Unmonitored exception received.

JVAB537
Java shell already active in job.

JVAB538
Error occurred when running Java shell.

JVAB539
Unable to start system debugger.

JVAB53A
Unable to start Java shell, reason code &1.

JVAB53B
Java processing canceled.

JVAB53D
Java Development Kit could not be found.

JVAB546
Error detected while running java in batch mode.

Top

Run Java Program (RUNJVA)

Where allowed to run: All environments (*ALL)
Threadsafe: No

Parameters
Examples
Error messages

The Run Java Program (JAVA) command runs the Java program associated with the specified Java class. If no Java program exists, one is created and associated with the class file.

This command can operate on files in any file system that supports the integrated file system APIs.

Top

Parameters

Keyword	Description	Choices	Notes
CLASS	Class file or JAR file	*Path name*, *VERSION	Required, Positional 1
PARM	Parameters	Single values: ***NONE** Other values (up to 200 repetitions): *Character value*	Optional, Positional 2
CLASSPATH	Classpath	*Path name*, ***ENVVAR**	Optional
CHKPATH	Classpath security check level	***WARN**, *SECURE, *IGNORE	Optional
OPTIMIZE	Optimization	***JIT**, *INTERPRET, 10, 20, 30, 40	Optional
INTERPRET	Interpret	***OPTIMIZE**, *YES, *NO, *JIT	Optional
PROP	Properties	Single values: ***NONE** Other values (up to 100 repetitions): *Element list*	Optional
	Element 1: Property name	*Path name*	
	Element 2: Property value	*Character value*, ***NONE**	
GCHINL	Garbage collect initial size	256-240000000, ***DFT**	Optional
GCHMAX	Garbage collect maximum size	256-240000000, ***DFT**, *NOMAX	Optional
GCFRQ	Garbage collection frequency	0-100, **50**	Optional
GCPTY	Garbage collection priority	**20**, 10, 30	Optional
OPTION	Option	Values (up to 4 repetitions): ***NONE**, *VERBOSE, *DEBUG, *VERBOSEGC, *NOCLASSGC	Optional
JOB	Job name	*Name*, **QJVACMDSRV**, *GEN	Optional
AGTPGM	Agent service program	Single values: ***NONE** Other values: *Qualified object name*	Optional
	Qualifier 1: Agent service program	*Name*	
	Qualifier 2: Library	*Name*, ***LIBL**, *CURLIB	
AGTOPTIONS	Agent options	*Character value*, ***NONE**	Optional
OUTPUT	Output	Single values: *PRINT, *NONE Other values: *Element list*	Optional
	Element 1: Destination	*****	
	Element 2: Program end action	***PAUSE**, *CONTINUE	

Top

Class file or JAR file (CLASS)

Specifies the class name or jar file to be run. The class name may be qualified by one or more package names. Each package name must be followed by a period. For example, CLASS('pkg1.pkg2.myclass') identifies a class qualified by two package names.

A jar file name may be specified only when running JDK 1.2 or higher. The start up class must be indicated by the Main-Class in the manifest header.

class-name
Specify the name of the class to be run.

jar-name
Specify the name of the jar file with the Main-Class specified in the manifest.

***VERSION**
The build version information for the Java Development Kit (JDK) and the Java Virtual Machine (JVM) is displayed. No Java program is run.

Top

Parameters (PARM)

Specifies one or more parameter values that are passed to the Java program. A maximum of 200 parameter values can be passed.

***NONE**
There are no input parameters to the Java program.

parameter-value
Specify the parameter value to be passed to the Java program.

Top

Classpath (CLASSPATH)

Specifies the path used to locate classes. Directories are separated by colons.

***ENVVAR**
The class path is determined by the environment variable CLASSPATH.

class-path
Path used to locate classes. An example class path is '/directory1/directory2:/QIBM/ProdData/Java400'.

Top

Classpath security check level (CHKPATH)

Specifies the level of warnings given for directories in the CLASSPATH that have public write authority. A directory in the CLASSPATH that has public write authority is a security exposure because it may contain a class file with the same name as the one you want to run. Whichever class file is found first is run.

***WARN**
A warning message is sent for each directory in the CLASSPATH that has public write authority.

***SECURE**
A warning message is sent for each directory in the CLASSPATH that has public write authority. If one or more warning messages are sent, an escape message is sent and the Java program is not run.

***IGNORE**
Ignore the fact that directories in the CLASSPATH may have public write authority. No warnings messages are sent.

Top

Optimization (OPTIMIZE)

Specifies how to treat class files if no Java program is associated with the file.

For 10,20,30,40 this specifies the optimization level of the Java program which will be created if no Java program is associated with the Java class file. The Java program will contain machine instruction sequences that are run when the Java program is invoked and will remain associated with the class file after the Java program has been run.

If the Java class file is determined to be downlevel or out of date, a new Java program will be created using the optimization level that was used when the previous Java program was created, instead of the value specified for this parameter.

For OPTIMIZE(*INTERPRET), the resulting Java program interprets the class byte codes when invoked.

OPTIMIZE(*INTERPRET) Java programs will be smaller but will run slower than Java programs created with higher optimization levels. As you increase the optimization level beyond 10, the Java program performance will generally improve, but the time required to create the Java program will increase and you will have less ability to debug the Java program.

For OPTIMIZE(*JIT), no Java program containing machine instruction sequences is created if no program is associated with the class file. Rather, the class is run using the Just In Time compiler (JIT).

***JIT** No Java program containing machine instruction sequences is created. The class is run using the Just In Time compiler (JIT).

***INTERPRET**
The Java program created does not contain machine specific instructions. It will be interpreted when the program is started. Variables can be displayed and modified while debugging.

If *OPTIMIZE is specified for the **Interpret (INTERPRET)** parameter, all of the classes that run will be run interpreted even if there is an optimized Java program associated with the class.

10 The Java program contains a compiled version of the class file byte codes and has only minimal additional compiler optimization. Variables can be displayed and modified while debugging.

20 The Java program contains a compiled version of the class file byte codes and has some additional compiler optimization. Variables can be displayed but not modified while debugging.

30 The Java program contains a compiled version of the class file byte codes and has more compiler optimization than optimization level 20. During a debug session, user variables cannot be changed, but can be displayed. The presented values may not be the current values of the variables.

40 The Java program contains a compiled version of the class file byte codes and has more compiler optimization than optimization level 30. All call and instruction tracing is disabled.

Top

Interpret (INTERPRET)

Specifies how the Java class files should be run.

***OPTIMIZE**
Whether all Java classes are run interpretively depends on the value specified for the OPTIMIZE parameter. If OPTIMIZE(*INTERPRET) was specified, all Java classes will be run interpretively. If any other value was specified for the OPTIMIZE parameter, only Java classes with Java programs created using CRTJVAPGM command and specifying OPTIMIZE(*INTERPRET) will be run interpretively.

***YES** All Java classes will be run interpretively regardless of the OPTIMIZE value associated Java program. Java classes that need a Java program created will use the optimization level specified in the OPTIMIZE parameter.

***NO** Only Java classes with Java programs created using CRTJVAPGM command and specifying OPTIMIZE(*INTERPRET) will be run interpretively. Java classes that need a Java program created will be created with the optimization level specified in the OPTIMIZE parameter.

***JIT** All Java class files will be run using the Just In Time compiler (JIT), regardless of what OPTIMIZE value was used when the associated Java program was created.

Top

Properties (PROP)

Specifies a list of values to assign to Java properties. Up to 100 properties can have a value assigned.

Single values

***NONE**
No properties are specified.

Element 1: Property name

property-name
Specify the name of the property to be changed.

Element 2: Property value

property-value
Specify the value to be assigned to the property.

Top

Garbage collect initial size (GCHINL)

Specifies the initial size, in kilobytes, of the garbage collection heap. This is used to prevent garbage collection from starting on small programs.

***DFT** The default initial size is 2048 kilobytes unless it is overridden by a property.

256-240000000
Specify the initial size, in kilobytes, of the garbage collection heap. It is recommended that the initial heap size be set to 2048 kilobytes (the default) or larger.

Top

Garbage collect maximum size (GCHMAX)

Specifies the maximum size, in kilobytes, that the garbage collection heap can grow to. This is used to prevent runaway programs that consume all of the available storage. Normally, garbage collection runs as an asynchronous thread in parallel with other threads. If the maximum size is reached, all other threads are stopped while garbage collection takes place. This may adversely affect performance.

***DFT** The default for the parameter is used. The default maximum is determined by the system unless a property is specified. The heap will grow until all system resources are depleted. Then a synchronous garbage collection is started to reclaim resources no longer in use.

***NOMAX**
The maximum size is not specified by the user. The maximum is determined by the system. The heap will grow until all system resources are depleted. Then a synchronous garbage collection is started to reclaim resources no longer in use.

256-240000000
Specify the size, in kilobytes, that the garbage collection heap can grow to.

Top

Garbage collection frequency (GCFRQ)

Specifies the relative frequency that garbage collection runs.

This parameter is no longer supported. It exists solely for compatibility with the releases earlier than Version 4 Release 3 Modification 0 of the server.

Top

Garbage collection priority (GCPTY)

Specifies the priority of the tasks running garbage collection.

This parameter is no longer supported. It exists solely for compatibility with the releases earlier than Version 4 Release 3 Modification 0 of the server.

Top

Option (OPTION)

Specifies special options used when running the Java class.

***NONE**
No special options are used when running the Java class.

***VERBOSE**
A message is displayed each time a class file is loaded.

***DEBUG**
Allows the system debugger to be used for this Java program. This value cannot be used with the JIT compiler. That is, *DEBUG is mutually exclusive with INTERPRET(*JIT). Also, *DEBUG cannot be used when a JAR file is specified for the CLASS keyword.

***VERBOSEGC**
A message is displayed for each garbage collection sweep.

***NOCLASSGC**
Unused classes are not reclaimed when garbage collection is run.

Top

Job name (JOB)

Specifies the name that is associated with the batch immediate (BCI) job that is started when this command is run. The BCI job is where the Java program will be run.

QJVACMDSRV
The job name for the BCI job will be QJVACMDSRV.

***GEN** The job name is generated from the class name.

name Specify the name to be used for the BCI job that is used to run the Java program.

Top

Agent service program (AGTPGM)

Specifies an ILE service program or OS/400 PASE program that contains a virtual machine (VM) agent. For example, a VM agent can be a remote debugger or profiler. The VM loads the agent program and looks for the entry point:

```
jint JNICALL JVM_OnLoad(JavaVM *jvm, char *options, void *reserved);
```

The VM calls the JVM_OnLoad function, passing a pointer to the JavaVM instance as the first argument, and the character string specified in the AGTOPTIONS parameter as the second argument. The third argument to JVM_OnLoad is reserved and set to NULL.

Single values

***NONE**
No VM agent program is specified.

Qualifier 1: Agent service program

name Specify the name of the VM agent program.

Qualifier 2: Library

***LIBL** All libraries in the library list for the current thread are searched until the first match is found. If the VM agent is an OS/400 PASE program, the directories contained in the LIBPATH and PASE_LIBPATH environment variables are used to locate the program.

***CURLIB**
The current library for the thread is searched. If no library is specified as the current library for the thread, the QGPL library is searched.

name Specify the name of the library to be searched.

Top

Agent options (AGTOPTIONS)

Specifies a character string of the virtual machine (VM) agent options. A pointer to this character string will be passed as the second argument to the JVM_OnLoad function in the VM agent program specified in the AGTPGM parameter. The format of the options string is determined by the agent program.

***NONE**
No options are specified. A NULL options argument will be passed to the JVM_OnLoad function.

character-value
Specify the options string to be passed to the VM agent program on start-up.

Top

Output (OUTPUT)

Specifies where output from the Java program should be sent and, if output is directed to the Java shell display, whether the shell display panel should go away when the Java program ends.

Single values

***PRINT**
The Java program output is sent to a spooled file through the QPRINT printer device file.

***NONE**
The Java program output is discarded.

Element 1: Destination

*****
A Java shell display panel will be used to display output if the Java program is run from an interactive job. If the Java program is run in a batch job, the Java program output is sent to a spooled file through the QPRINT printer device file.

Element 2: Program end action

***PAUSE**
The Java shell display panel is shown until the F3, F12, or Enter key is pressed.

***CONTINUE**
The Java shell display panel is closed (goes away) when the Java program ends.

Top

Examples

Example 1: Run a Java Program

```
JAVA   CLASS('projectA.myJavaclassname')
```

This command will run the iSeries Java program associated with the class *myJavaclassname*. The job name of the batch immediate (BCI) job where the Java program will run will be QJVACMDSRV.

Example 2: Generate the Job Name for the Java Program

```
JAVA   CLASS('projectA.myJavaclassname') JOB(*GEN)
```

This command will run the iSeries Java program associated with the class *myJavaclassname*. The job name of the batch immediate (BCI) job where the Java program will run will be MYJAVACLAS.

Top

Error messages

***ESCAPE Messages**

JVAB534
Unable to complete Java program ″&1″.

JVAB535
Unmonitored exception received.

JVAB537
Java shell already active in job.

JVAB538
Error occurred when running Java shell.

JVAB539
Unable to start system debugger.

JVAB53A
Unable to start Java shell, reason code &1.

JVAB53B
Java processing canceled.

JVAB53D
Java Development Kit could not be found.

JVAB546
Error detected while running java in batch mode.

Top

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may 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 may have patents or pending patent applications covering subject matter described 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
500 Columbus Avenue
Thornwood, NY8809
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 publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication 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

Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
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 information 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.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. 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. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. 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 IBM's application programming interfaces.

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

Trademarks

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

Advanced Function Printing
AFP
AS/400
CICS
COBOL/400
C/400
DataPropagator
DB2
IBM
Infoprint
InfoWindow
iSeries
LPDA
OfficeVision

OS/400
Print Services Facility
RPG/400
SystemView
System/36
TCS
WebSphere

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Terms and conditions for downloading and printing publications

Permissions for the use of the publications you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

Personal Use: You may reproduce these Publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these Publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

All material copyrighted by IBM Corporation.

By downloading or printing a publication from this site, you have indicated your agreement with these terms and conditions.

Code disclaimer information

This document contains programming examples.

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

All sample code is provided by IBM for illustrative purposes only. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

All programs contained herein are provided to you ″AS IS″ without any warranties of any kind. The implied warranties of non-infringement, merchantability and fitness for a particular purpose are expressly disclaimed.

Printed in USA