Rocky Enterprise Linux 9.2 Manual Pages on command 'rmic-java-11-openjdk-11.0.20.0.8-3.el9.x86_64.1'

$ man rmic-java-11-openjdk-11.0.20.0.8-3.el9.x86_64.1

rmic(1) Remote Method Invocation (RMI) Tools rmic(1)

NAME

rmic - Generates stub, skeleton, and tie classes for remote objects that use the Java Remote Method Protocol (JRMP) or Internet Inter-Orb protocol (IIOP). Also generates Object Management Group (OMG) Interface Definition Language (IDL)

SYNOPSIS

rmic [options] package-qualified-class-names

options

The command-line options. See Options.

package-qualified-class-names

Class names that include their packages, for example, java.awt.Color.

DESCRIPTION

Deprecation Note: Support for static generation of Java Remote Method Protocol (JRMP) stubs and skeletons has been deprecated. Oracle recommends that you use dynamically generated JRMP stubs instead, eliminating the need to use this tool for JRMP-based applications. See

the java.rmi.server.UnicastRemoteObject specification at http://docs.oracle.com/javase/8/docs/api/java/rmi/server/UnicastRemoteObject.html for further information.

The rmic compiler generates stub and skeleton class files using the Java Remote Method Protocol (JRMP) and stub and tie class files (IIOP protocol) for remote objects. These class files are generated from compiled Java programming language classes that are remote object implementation classes. A remote implementation class is a class that implements the interface java.rmi.Remote. The class names in the rmic command must be for classes that were compiled successfully with the javac command and must be fully package qualified. For example, running the rmic command on the class file name HelloImpl as shown here creates the HelloImpl_Stub.classfile in the hello subdirectory (named for the class's package):

rmic hello.HelloImpl

A skeleton for a remote object is a JRMP protocol server-side entity that has a method that dispatches calls to the remote object implementation.

A tie for a remote object is a server-side entity similar to a skeleton, but communicates with the client with the IIOP protocol.

A stub is a client-side proxy for a remote object that is responsible for communicating method invocations on remote objects to the server where the actual remote object implementation resides. A client's reference to a remote object, therefore, is actually a reference to a local stub.

By default, the rmic command generates stub classes that use the 1.2 JRMP stub protocol version only, as though the -v1.2 option was specified. The -vcompat option was the default in releases before 5.0.

Use the -iiop option to generate stub and tie classes for the IIOP protocol. See Options.

A stub implements only the remote interfaces, and not any local interfaces that the remote object also implements. Because a JRMP stub implements the same set of remote interfaces as the remote object, a

client can use the Java programming language built-in operators for casting and type checking. For IIOP, the PortableRemoteObject.narrow method must be used.

OPTIONS

-bootclasspath path

Overrides the location of bootstrap class files.

-classpath path

Specifies the path the rmic command uses to look up classes. This option overrides the default or the CLASSPATH environment variable when it is set. Directories are separated by colons. The general format for path is: .:<your_path>, for example: .:/usr/local/java/classes.

-d directory

Specifies the root destination directory for the generated class hierarchy. You can use this option to specify a destination directory for the stub, skeleton, and tie files. For example, the following command places the stub and skeleton classes derived from MyClass into the directory /java/classes/exampleclass.

rmic -d /java/classes exampleclass.MyClass

If the -d option is not specified, then the default behavior is as if -d . was specified. The package hierarchy of the target class is created in the current directory, and stub/tie/skeleton files are placed within it. In some earlier releases of the rmic command, if the -d option was not specified, then the package hierarchy was not created, and all of the output files were placed directly in the current directory.

-extdirs path

Overrides the location of installed extensions.

-g

Enables the generation of all debugging information, including local variables. By default, only line number information is generated.

-idl

Causes the rmic command to generate OMG IDL for the classes specified and any classes referenced. IDL provides a purely declarative, programming language-independent way to specify an API for an object. The IDL is used as a specification for methods and data that can be written in and called from any language that provides CORBA bindings. This includes Java and C++ among others. See Java IDL: IDL to Java Language Mapping at http://docs.oracle.com/javase/8/docs/technotes/guides/idl/mapping/jidlMapping.html

When the -idl option is used, other options also include:

? The -always or -alwaysgenerate options force regeneration even when existing stubs/ties/IDL are newer than the input class.

? The -factory option uses the factory keyword in generated IDL.

? The -idlModule from JavaPackage[.class]toIDLModule specifies IDLEntity package mapping, for example: -idlModulemy.module my::real::idlmod.

? -idlFilefromJavaPackage[.class] toIDLFile specifies IDLEntity file mapping, for example: -idlFile test.pkg.X TEST16.idl.

-iiop

Causes the rmic command to generate IIOP stub and tie classes, rather than JRMP stub and skeleton classes. A stub class is a local proxy for a remote object and is used by clients to send calls to a server. Each remote interface requires a stub class, which implements that remote interface. A client reference to a remote object is a reference to a stub. Tie classes are used on the server side to process incoming calls, and dispatch the calls to the proper implementation class. Each implementation class requires a tie class.

If you call the rmic command with the -iiop, then it generates stubs and ties that conform to this naming convention:

_<implementationName>_stub.class

_<interfaceName>_tie.class

? When you use the -iiop option, other options also include:

? The -always or -alwaysgenerate options force regeneration even when existing stubs/ties/IDL are newer than the input class.

? The -nolocalstubs option means do not create stubs optimized for same-process clients and servers.

? The -noValueMethods option must be used with the -idl option. The -noValueMethods option prevents the addition of valuetype methods and initializers to emitted IDL. These methods and initializers are optional for valuetypes, and are generated unless the -noValueMethods option is specified with the -idl option.

? The -poa option changes the inheritance from org.omg.CORBA_2_3.portable.ObjectImpl to org.omg.PortableServer.Servant. The PortableServer module for the Portable Object Adapter (POA) defines the native Servant type. In the Java programming language, the Servant type is mapped to the Java org.omg.PortableServer.Servant class. It serves as the base class for all POA servant implementations and provides a number of methods that can be called by the application programmer, and methods that are called by the POA and that can be overridden by the user to control aspects of servant behavior. Based on the OMG IDL to Java Language Mapping Specification, CORBA V 2.3.1 ptc/00-01-08.pdf..RE

-J

Used with any Java command, the -J option passes the argument that follows the -J (no spaces between the -Jand the argument) to the Java interpreter

-keep or -keepgenerated

Retains the generated .java source files for the stub, skeleton, and tie classes and writes them to the same directory as the.class files.

-nowarn

Turns off warnings. When the -nowarn options is used. The compiler does not print out any warnings.

-nowrite

Does not write compiled classes to the file system.

-vcompat (deprecated)

Generates stub and skeleton classes that are compatible with both the 1.1 and 1.2 JRMP stub protocol versions. This option was the default in releases before 5.0. The generated stub classes use the 1.1 stub protocol version when loaded in a JDK 1.1 virtual machine and use the 1.2 stub protocol version when loaded into a 1.2 (or later) virtual machine. The generated skeleton classes support both 1.1 and 1.2 stub protocol versions. The generated classes are relatively large to support both modes of operation. Note: This option has been deprecated. See Description.

-verbose

Causes the compiler and linker to print out messages about what classes are being compiled and what class files are being loaded.

-v1.1 (deprecated)

Generates stub and skeleton classes for the 1.1 JRMP stub protocol version only. The -v1.1 option is only useful for generating stub classes that are serialization-compatible with preexisting, statically deployed stub classes that were generated by the rmic command from JDK 1.1 and that cannot be upgraded (and dynamic class loading is not being used). Note: This option has been deprecated. See Description.

-v1.2 (deprecated)

(Default) Generates stub classes for the 1.2 JRMP stub protocol version only. No skeleton classes are generated because skeleton classes are not used with the 1.2 stub protocol version. The generated stub classes do not work when they are loaded into a JDK 1.1 virtual machine. Note: This option has been deprecated. See Description.

CLASSPATH

Used to provide the system a path to user-defined classes.

Directories are separated by colons, for example:

.:/usr/local/java/classes.

SEE ALSO

? javac(1)

? java(1)

? Setting the Class Path