



Rocky Enterprise Linux 9.2 Manual Pages on command 'jdeps.1'

\$ man jdeps.1

jdeps(1) Basic Tools jdeps(1)

NAME

jdeps - Java class dependency analyzer.

SYNOPSIS

jdeps [options] classes ...

options

Command-line options. See Options.

classes

Name of the classes to analyze. You can specify a class that can be found in the class path, by its file name, a directory, or a JAR file.

DESCRIPTION

The jdeps command shows the package-level or class-level dependencies of Java class files. The input class can be a path name to a .class file, a directory, a JAR file, or it can be a fully qualified class name to analyze all class files. The options determine the output. By default, jdeps outputs the dependencies to the system output. It can generate the dependencies in DOT language (see the -dotoutput option).

OPTIONS

`-dotoutput <dir>`

Destination directory for DOT file output. If specified, `jdeps` will generate one dot file per each analyzed archive named `<archive-file-name>.dot` listing the dependencies, and also a summary file named `summary.dot` listing the dependencies among the archives.

`-s, -summary`

Prints dependency summary only.

`-v, -verbose`

Prints all class-level dependencies.

`-verbose:package`

Prints package-level dependencies excluding dependencies within the same archive.

`-verbose:class`

Prints class-level dependencies excluding dependencies within the same archive.

`-cp <path>, -classpath <path>`

Specifies where to find class files.

See also [Setting the Class Path](#).

`-p <pkg name>, -package <pkg name>`

Finds dependencies in the specified package. You can specify this option multiple times for different packages. The `-p` and `-e` options are mutually exclusive.

`-e <regex>, -regex <regex>`

Finds dependencies in packages matching the specified regular expression pattern. The `-p` and `-e` options are mutually exclusive.

`-include <regex>`

Restricts analysis to classes matching pattern. This option filters the list of classes to be analyzed. It can be used together with `-p` and `-e` which apply pattern to the dependencies.

`-jdkinternals`

Finds class-level dependences in JDK internal APIs. By default, it analyzes all classes specified in the -classpath option and in input files unless you specified the -include option. You cannot use this option with the -p, -e, and -s options.

Warning: JDK internal APIs may not be accessible in upcoming releases.

-P, -profile

Shows profile or the file containing a package.

-apionly

Restricts analysis to APIs, for example, dependences from the signature of public and protected members of public classes including field type, method parameter types, returned type, and checked exception types.

-R, -recursive

Recursively traverses all dependencies.

-version

Prints version information.

-h, -?, -help

Prints help message for jdeps.

EXAMPLES

Analyzing the dependencies of Notepad.jar.

```
$ jdeps demo/jfc/Notepad/Notepad.jar
```

```
demo/jfc/Notepad/Notepad.jar -> /usr/java/jre/lib/rt.jar
```

```
<unnamed> (Notepad.jar)
```

```
-> java.awt
```

```
-> java.awt.event
```

```
-> java.beans
```

```
-> java.io
```

```
-> java.lang
```

```
-> java.net
```

```
-> java.util
```

```
-> java.util.logging
```

```
-> javax.swing
```

- > javax.swing.border
- > javax.swing.event
- > javax.swing.text
- > javax.swing.tree
- > javax.swing.undo

Use -P or -profile option to show on which profile that Notepad depends.

```
$ jdeps -profile demo/jfc/Notepad/Notepad.jar
```

```
demo/jfc/Notepad/Notepad.jar -> /usr/java/jre/lib/rt.jar (Full JRE)
```

```
<unnamed> (Notepad.jar)
```

-> java.awt	Full JRE
-> java.awt.event	Full JRE
-> java.beans	Full JRE
-> java.io	compact1
-> java.lang	compact1
-> java.net	compact1
-> java.util	compact1
-> java.util.logging	compact1
-> javax.swing	Full JRE
-> javax.swing.border	Full JRE
-> javax.swing.event	Full JRE
-> javax.swing.text	Full JRE
-> javax.swing.tree	Full JRE
-> javax.swing.undo	Full JRE

Analyzing the immediate dependencies of a specific class in a given classpath, for example the com.sun.tools.jdeps.Main class in the tools.jar file.

```
$ jdeps -cp lib/tools.jar com.sun.tools.jdeps.Main
```

```
lib/tools.jar -> /usr/java/jre/lib/rt.jar
```

```
com.sun.tools.jdeps (tools.jar)
```

- > java.io
- > java.lang

Use the -verbose:class option to find class-level dependencies or use

the -v or -verbose option to include dependencies from the same JAR file.

```
$ jdeps -verbose:classpath -cp lib/tools.jar com.sun.tools.jdeps.Main
```

```
lib/tools.jar -> /usr/java/jre/lib/rt.jar
```

```
com.sun.tools.jdeps.Main (tools.jar)
```

```
-> java.io.PrintWriter
```

```
-> java.lang.Exception
```

```
-> java.lang.Object
```

```
-> java.lang.String
```

```
-> java.lang.System
```

Use the -R or -recursive option to analyze the transitive dependencies of the com.sun.tools.jdeps.Main class.

```
$ jdeps -R -cp lib/tools.jar com.sun.tools.jdeps.Main
```

```
lib/tools.jar -> /usr/java/jre/lib/rt.jar
```

```
com.sun.tools.classfile (tools.jar)
```

```
-> java.io
```

```
-> java.lang
```

```
-> java.lang.reflect
```

```
-> java.nio.charset
```

```
-> java.nio.file
```

```
-> java.util
```

```
-> java.util.regex
```

```
com.sun.tools.jdeps (tools.jar)
```

```
-> java.io
```

```
-> java.lang
```

```
-> java.nio.file
```

```
-> java.nio.file.attribute
```

```
-> java.text
```

```
-> java.util
```

```
-> java.util.jar
```

```
-> java.util.regex
```

```
-> java.util.zip
```

```
/usr/java/jre/lib/jce.jar -> /usr/java/jre/lib/rt.jar
```

javax.crypto (jce.jar)

- > java.io
- > java.lang
- > java.lang.reflect
- > java.net
- > java.nio
- > java.security
- > java.security.cert
- > java.security.spec
- > java.util
- > java.util.concurrent
- > java.util.jar
- > java.util.regex
- > java.util.zip
- > javax.security.auth
- > sun.security.jca JDK internal API (rt.jar)
- > sun.security.util JDK internal API (rt.jar)

javax.crypto.spec (jce.jar)

- > java.lang
- > java.security.spec
- > java.util

/usr/java/jre/lib/rt.jar -> /usr/java/jre/lib/jce.jar

java.security (rt.jar)

- > javax.crypto

Generate dot files of the dependencies of Notepad demo.

\$ jdeps -dotoutput dot demo/jfc/Notepad/Notepad.jar

jdeps will create one dot file for each given JAR file named

<filename>.dot in the dot directory specified in the -dotoutput option,

and also a summary file named summary.dot that will list the

dependencies among the JAR files

\$ cat dot/Notepad.jar.dot

digraph "Notepad.jar" {

// Path: demo/jfc/Notepad/Notepad.jar

```

"<unnamed>"          -> "java.awt";
"<unnamed>"          -> "java.awt.event";
"<unnamed>"          -> "java.beans";
"<unnamed>"          -> "java.io";
"<unnamed>"          -> "java.lang";
"<unnamed>"          -> "java.net";
"<unnamed>"          -> "java.util";
"<unnamed>"          -> "java.util.logging";
"<unnamed>"          -> "javax.swing";
"<unnamed>"          -> "javax.swing.border";
"<unnamed>"          -> "javax.swing.event";
"<unnamed>"          -> "javax.swing.text";
"<unnamed>"          -> "javax.swing.tree";
"<unnamed>"          -> "javax.swing.undo";
}
$ cat dot/summary.dot
digraph "summary" {
    "Notepad.jar"      -> "rt.jar";
}

```

SEE ALSO

? javap(1)

JDK 8

21 November 2013

jdeps(1)