NAME

jps – Lists the instrumented Java Virtual Machines (JVMs) on the target system. This command is experimental and unsupported.

SYNOPSIS

jps [*options*] [*hostid*]

options	Command-line options. See Options.
hostid	The identifier of the host for which the process report should be generated. The **hostid** can include optional components that indicate the communications protocol, port number, and other implementation specific data. See Host Identifier.

DESCRIPTION

The **jps** command lists the instrumented Java HotSpot VMs on the target system. The command is limited to reporting information on JVMs for which it has the access permissions.

If the **jps** command is run without specifying a **hostid**, then it searches for instrumented JVMs on the local host. If started with a **hostid**, then it searches for JVMs on the indicated host, using the specified protocol and port. A **jstatd** process is assumed to be running on the target host.

The **jps** command reports the local JVM identifier, or **lvmid**, for each instrumented JVM found on the target system. The **lvmid** is typically, but not necessarily, the operating system's process identifier for the JVM process. With no options, **jps** lists each Java application's **lvmid** followed by the short form of the application's class name or jar file name. The short form of the class name or JAR file name omits the class's package information or the JAR files path information.

The **jps** command uses the Java launcher to find the class name and arguments passed to the main method. If the target JVM is started with a custom launcher, then the class or JAR file name and the arguments to the **main** method are not available. In this case, the **jps** command outputs the string **Unknown** for the class name or JAR file name and for the arguments to the **main** method.

The list of JVMs produced by the **jps** command can be limited by the permissions granted to the principal running the command. The command only lists the JVMs for which the principle has access rights as determined by operating system-specific access control mechanisms.

OPTIONS

The **jps** command supports a number of options that modify the output of the command. These options are subject to change or removal in the future.

-q

Suppresses the output of the class name, JAR file name, and arguments passed to the **main** method, producing only a list of local JVM identifiers.

-m

Displays the arguments passed to the **main** method. The output may be **null** for embedded JVMs.

-l

Displays the full package name for the application's **main** class or the full path name to the application's JAR file.

-v

Displays the arguments passed to the JVM.

-V

Suppresses the output of the class name, JAR file name, and arguments passed to the main method, producing only a list of local JVM identifiers.

-J**option**

Passes **option** to the JVM, where option is one of the **options** described on the reference page for the Java application launcher. For example, **-J-Xms48m** sets the startup memory to 48 MB. See java(1).

HOST IDENTIFIER

The host identifier, or **hostid** is a string that indicates the target system. The syntax of the **hostid** string corresponds to the syntax of a URI:

```
[protocol:][[//]hostname][:port][/servername]
```

protocol
: The communications protocol. If the **protocol** is omitted and a **hostname** is not specified, then the default protocol is a platform-specific, optimized, local protocol. If the protocol is omitted and a host name is specified, then the default protocol is **rmi**.

hostname
: A hostname or IP address that indicates the target host. If you omit the **hostname** parameter, then the target host is the local host.

port
: The default port for communicating with the remote server. If the **hostname** parameter is omitted or the **protocol** parameter specifies an optimized, local protocol, then the **port** parameter is ignored. Otherwise, treatment of the **port** parameter is implementation specific. For the default **rmi** protocol, the **port** parameter indicates the port number for the rmiregistry on the remote host. If the **port** parameter is omitted, and the **protocol** parameter indicates **rmi**, then the default rmiregistry port (1099) is used.

servername
: The treatment of this parameter depends on the implementation. For the optimized, local protocol, this field is ignored. For the **rmi** protocol, this parameter is a string that represents the name of the RMI remote object on the remote host. See the **jstatd** command **-n**option for more information.

OUTPUT FORMAT

The output of the **jps** command follows the following pattern:

```
lvmid [ [ classname | JARfilename | "Unknown"] [ arg* ] [ jvmarg* ] ]
```

All output tokens are separated by white space. An **arg** value that includes embedded white space introduces ambiguity when attempting to map arguments to their actual positional parameters.

Note: It is recommended that you do not write scripts to parse **jps** output because the format might change in future releases. If you write scripts that parse **jps** output, then expect to modify them for future releases of this tool.

EXAMPLES

This section provides examples of the **jps** command.

List the instrumented JVMs on the local host:

```
jps
18027 Java2Demo.JAR
18032 jps
18005 jstat
```

The following example lists the instrumented JVMs on a remote host. This example assumes that the **jstat** server and either the its internal RMI registry or a separate external rmiregistry process are running on the remote host on the default port (port 1099). It also assumes that the local host has appropriate permissions to access the remote host. This example also includes the **-l** option to output the long form of the class names or JAR file names.

```
jps –l remote.domain
3002 /opt/jdk1.7.0/demo/jfc/Java2D/Java2Demo.JAR
2857 sun.tools.jstatd.jstatd
```

The following example lists the instrumented JVMs on a remote host with a non-default port for the RMI registry. This example assumes that the **jstatd** server, with an internal RMI registry bound to port 2002, is running on the remote host. This example also uses the **-m** option to include the arguments passed to the **main** method of each of the listed Java applications.

```
jps −m remote.domain:2002
3002 /opt/jdk1.7.0/demo/jfc/Java2D/Java2Demo.JAR
3102 sun.tools.jstatd.jstatd −p 2002
```

SEE ALSO

- java(1)
- jstat(1)
- jstatd(1)
- rmiregistry(1)