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

$ man jstatd.1

jstatd(1) Monitoring Tools jstatd(1)

NAME

jstatd - Monitors Java Virtual Machines (JVMs) and enables remote monitoring tools to attach to JVMs. This command is experimental and unsupported.

SYNOPSIS

jstatd [options]

options

The command-line options. See Options.

DESCRIPTION

The jstatd command is an RMI server application that monitors for the creation and termination of instrumented Java HotSpot VMs and provides an interface to enable remote monitoring tools to attach to JVMs that are running on the local host.

The jstatd server requires an RMI registry on the local host. The jstatd server attempts to attach to the RMI registry on the default port, or on the port you specify with the -pport option. If an RMI registry is not found, then one is created within the jstatd

application that is bound to the port that is indicated by the -pport option or to the default RMI registry port when the -pport option is omitted. You can stop the creation of an internal RMI registry by specifying the -nr option.

OPTIONS

-nr

Does not attempt to create an internal RMI registry within the jstatd process when an existing RMI registry is not found.

-p port

The port number where the RMI registry is expected to be found, or when not found, created if the -nr option is not specified.

-n rminame

Name to which the remote RMI object is bound in the RMI registry. The default name is JStatRemoteHost. If multiple jstatd servers are started on the same host, then the name of the exported RMI object for each server can be made unique by specifying this option. However, doing so requires that the unique server name be included in the monitoring client's hostid and vmid strings.

-Joption

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).

SECURITY

The jstatd server can only monitor JVMs for which it has the appropriate native access permissions. Therefore, the jstatd process must be running with the same user credentials as the target JVMs. Some user credentials, such as the root user in UNIX-based systems, have permission to access the instrumentation exported by any JVM on the system. A jstatd process running with such credentials can monitor any JVM on the system, but introduces additional security concerns.

The jstatd server does not provide any authentication of remote

clients. Therefore, running a jstatd server process exposes the instrumentation export by all JVMs for which the jstatd process has access permissions to any user on the network. This exposure might be undesirable in your environment, and therefore, local security policies should be considered before you start the jstatd process, particularly in production environments or on networks that are not secure.

The jstatd server installs an instance of RMISecurityPolicy when no other security manager is installed, and therefore, requires a security policy file to be specified. The policy file must conform to Default Policy Implementation and Policy File Syntax at http://docs.oracle.com/javase/8/docs/technotes/guides/security/PolicyFiles.html

The following policy file allows the jstatd server to run without any security exceptions. This policy is less liberal than granting all permissions to all code bases, but is more liberal than a policy that grants the minimal permissions to run the jstatd server.

```
grant codebase "file:${java.home}/../lib/tools.jar" {
    permission java.security.AllPermission;
};
```

To use this policy setting, copy the text into a file called jstatd.all.policy and run the jstatd server as follows:

```
jstatd -J-Djava.security.policy=jstatd.all.policy
```

For sites with more restrictive security practices, it is possible to use a custom policy file to limit access to specific trusted hosts or networks, though such techniques are subject to IP address spoofing attacks. If your security concerns cannot be addressed with a customized policy file, then the safest action is to not run the jstatd server and use the jstat and jps tools locally.

REMOTE INTERFACE

The interface exported by the jstatd process is proprietary and guaranteed to change. Users and developers are discouraged from writing to this interface.

EXAMPLES

The following are examples of the jstatd command. The jstatd scripts

automatically start the server in the background

INTERNAL RMI REGISTRY

This example shows hos to start a jstatd session with an internal RMI registry. This example assumes that no other server is bound to the default RMI registry port (port 1099).

jstatd -J-Djava.security.policy=all.policy

EXTERNAL RMI REGISTRY

This example starts a jstatd session with a external RMI registry.

rmiregistry&

jstatd -J-Djava.security.policy=all.policy

This example starts a jstatd session with an external RMI registry server on port 2020.

jrmiregistry 2020&

jstatd -J-Djava.security.policy=all.policy -p 2020

This example starts a jstatd session with an external RMI registry on port 2020 that is bound to AlternateJstatdServerName.

rmiregistry 2020&

jstatd -J-Djava.security.policy=all.policy -p 2020

-n AlternateJstatdServerName

STOP THE CREATION OF AN IN-PROCESS RMI REGISTRY

This example starts a jstatd session that does not create an RMI registry when one is not found. This example assumes an RMI registry is already running. If an RMI registry is not running, then an error message is displayed.

jstatd -J-Djava.security.policy=all.policy -nr

ENABLE RMI LOGGING

This example starts a jstatd session with RMI logging capabilities enabled. This technique is useful as a troubleshooting aid or for monitoring server activities.

jstatd -J-Djava.security.policy=all.policy

-J-Djava.rmi.server.logCalls=true

SEE ALSO

? java(1)

? jps(1)

? jstat(1)

? rmiregistry(1)