NAME

rmid – Starts the activation system daemon that enables objects to be registered and activated in a Java Virtual Machine (JVM).

SYNOPSIS

rmid [*options*]

options   The command-line options. See Options.

DESCRIPTION

The **rmid** command starts the activation system daemon. The activation system daemon must be started before activatable objects can be either registered with the activation system or activated in a JVM. For details on how to write programs that use activatable objects, the *Using Activation* tutorial at http://docs.oracle.com/javase/8/docs/technotes/guides/rmi/activation/overview.html

Start the daemon by executing the **rmid** command and specifying a security policy file, as follows:

rmid −J−Djava.security.policy=rmid.policy

When you run Oracle's implementation of the **rmid** command, by default you must specify a security policy file so that the **rmid** command can verify whether or not the information in each **ActivationGroupDesc** is allowed to be used to start a JVM for an activation group. Specifically, the command and options specified by the **CommandEnvironment** and any properties passed to an **ActivationGroupDesc** constructor must now be explicitly allowed in the security policy file for the **rmid** command. The value of the **sun.rmi.activation.execPolicy** property dictates the policy that the **rmid** command uses to determine whether or not the information in an **ActivationGroupDesc** can be used to start a JVM for an activation group. For more information see the description of the -J-Dsun.rmi.activation.execPolicy=policy option.

Executing the **rmid** command starts the Activator and an internal registry on the default port1098 and binds an **ActivationSystem** to the name **java.rmi.activation.ActivationSystem** in this internal registry.

To specify an alternate port for the registry, you must specify the **-port** option when you execute the **rmid** command. For example, the following command starts the activation system daemon and a registry on the registry's default port, 1099.

rmid −J−Djava.security.policy=rmid.policy −port 1099

START RMID ON DEMAND

An alternative to starting **rmid** from the command line is to configure **inetd** (Oracle Solaris) or **xinetd** (Linux) to start **rmid** on demand.

When RMID starts, it attempts to obtain an inherited channel (inherited from **inetd**/**xinetd**) by calling the **System.inheritedChannel** method. If the inherited channel is null or not an instance of **java.nio.channels.ServerSocketChannel**, then RMID assumes that it was not started by **inetd**/**xinetd**, and it starts as previously described.

If the inherited channel is a **ServerSocketChannel** instance, then RMID uses the **java.net.ServerSocket** obtained from the **ServerSocketChannel** as the server socket that accepts requests for the remote objects it exports: The registry in which the **java.rmi.activation.ActivationSystem** is bound and the **java.rmi.activation.Activator** remote object. In this mode, RMID behaves the same as when it is started from the command line, except in the following cases:

- Output printed to **System.err** is redirected to a file. This file is located in the directory specified by the **java.io.tmpdir** system property (typically **/var/tmp** or **/tmp**) with the prefix **rmid-err** and the suffix **tmp**.

- The **-port** option is not allowed. If this option is specified, then RMID exits with an error message.

- The **-log** option is required. If this option is not specified, then RMID exits with an error message

See the man pages for **inetd** (Oracle Solaris) or **xinetd** (Linux) for details on how to configure services to be started on demand.

OPTIONS

-C*option*

Specifies an option that is passed as a command-line argument to each child process (activation group) of the **rmid** command when that process is created. For example, you could pass a property to each virtual machine spawned by the activation system daemon:

rmid −C−Dsome.property=value

This ability to pass command-line arguments to child processes can be useful for debugging. For example, the following command enables server-call logging in all child JVMs.

rmid −C−Djava.rmi.server.logCalls=true

-J*option*

Specifies an option that is passed to the Java interpreter running RMID. For example, to specify that the **rmid** command use a policy file named **rmid.policy**, the **-J** option can be used to define the **java.security.policy** property on the **rmid** command line, for example:

rmid −J−Djava.security.policy−rmid.policy

-J-Dsun.rmi.activation.execPolicy=*policy*

Specifies the policy that RMID employs to check commands and command-line options used to start the JVM in which an activation group runs. Please note that this option exists only in Oracle's implementation of the Java RMI activation daemon. If this property is not specified on the command line, then the result is the same as though **-J-Dsun.rmi.activation.execPolicy=default** were specified. The possible values of **policy** can be **default**, **policyClassName**, or **none**.

- default

  The **default** or unspecified value **execPolicy** allows the **rmid** command to execute commands with specific command-line options only when the **rmid** command was granted permission to execute those commands and options in the security policy file that the **rmid** command uses. Only the default activation group implementation can be used with the default execution policy.

  The **rmid** command starts a JVM for an activation group with the information in the group's registered activation group descriptor, an **ActivationGroupDesc**. The group descriptor specifies an optional **ActivationGroupDesc.CommandEnvironment** that includes the command to execute to start the activation group and any command-line options to be added to the command line. By default, the **rmid** command uses the **java** command found in **java.home**. The group descriptor also contains properties overrides that are added to the command line as options defined as: **-D<property>=<value>**.The **com.sun.rmi.rmid.ExecPermission** permission grants the **rmid** command permission to execute a command that is specified in the group descriptor's **CommandEnvironment** to start an activation group. The **com.sun.rmi.rmid.ExecOptionPermission** permission enables the **rmid** command to use command-line options, specified as properties overrides in the group

descriptor or as options in the **CommandEnvironment** when starting the activation group.When granting the **rmid** command permission to execute various commands and options, the permissions **ExecPermission** and **ExecOptionPermission** must be granted to all code sources.

ExecPermission

The **ExecPermission** class represents permission for the **rmid** command to execute a specific command to start an activation group.

Syntax: The name of an **ExecPermission** is the path name of a command to grant the **rmid** command permission to execute. A path name that ends in a slash (/) and an asterisk (*) indicates that all of the files contained in that directory where slash is the file-separator character, **File.separatorChar**. A path name that ends in a slash (/) and a minus sign (-) indicates all files and subdirectories contained in that directory (recursively). A path name that consists of the special token **<<ALL FILES>>** matches any file.

A path name that consists of an asterisk (*) indicates all the files in the current directory. A path name that consists of a minus sign (-) indicates all the files in the current directory and (recursively) all files and subdirectories contained in the current directory.

ExecOptionPermission

The **ExecOptionPermission** class represents permission for the **rmid** command to use a specific command-line option when starting an activation group. The name of an **ExecOptionPermission** is the value of a command-line option.

Syntax: Options support a limited wild card scheme. An asterisk signifies a wild card match, and it can appear as the option name itself (matches any option), or an asterisk (*) can appear at the end of the option name only when the asterisk (*) follows a dot (.) or an equals sign (=).

For example: ***** or **-Dmydir.*** or **-Da.b.c=*** is valid, but ***mydir** or **-Da*b** or **ab*** is not.

Policy file for rmid

When you grant the **rmid** command permission to execute various commands and options, the permissions **ExecPermission** and **ExecOptionPermission** must be granted to all code sources (universally). It is safe to grant these permissions universally because only the **rmid** command checks these permissions.

An example policy file that grants various execute permissions to the **rmid** command is:

```
grant {
    permission com.sun.rmi.rmid.ExecPermission
      "/files/apps/java/jdk1.7.0/solaris/bin/java";
    permission com.sun.rmi.rmid.ExecPermission
      "/files/apps/rmidcmds/*";
    permission com.sun.rmi.rmid.ExecOptionPermission
      "−Djava.security.policy=/files/policies/group.policy";
    permission com.sun.rmi.rmid.ExecOptionPermission
      "−Djava.security.debug=*";
    permission com.sun.rmi.rmid.ExecOptionPermission
      "−Dsun.rmi.*";
};
```

The first permission granted allows the **rmid** tcommand o execute the 1.7.0 release of the **java** command, specified by its explicit path name. By default, the version of the **java** command found in **java.home** is used (the same one that the **rmid** command uses), and does not need to be specified in the policy file. The second permission allows the **rmid** command to execute any command in the directory **/files/apps/rmidcmds**.

The third permission granted, an **ExecOptionPermission**, allows the **rmid** command to start an activation group that defines the security policy file to be **/files/policies/group.policy**. The next permission allows the **java.security.debug property** to be used by an activation group. The last permission allows any property in the **sun.rmi property** name hierarchy to be used by activation groups.

To start the **rmid** command with a policy file, the **java.security.policy** property needs to be specified on the **rmid** command line, for example:

rmid -J-Djava.security.policy=rmid.policy.

*   <policyClassName>

    If the default behavior is not flexible enough, then an administrator can provide, when starting the **rmid** command, the name of a class whose **checkExecCommand** method is executed to check commands to be executed by the **rmid** command.

    The **policyClassName** specifies a public class with a public, no-argument constructor and an implementation of the following **checkExecCommand** method:

    public void checkExecCommand(ActivationGroupDesc desc, String[] command)
    throws SecurityException;

    Before starting an activation group, the **rmid** command calls the policy's **checkExecCommand** method and passes to it the activation group descriptor and an array that contains the complete command to start the activation group. If the **checkExecCommand** throws a **SecurityException**, then the **rmid** command does not start the activation group and an **ActivationException** is thrown to the caller attempting to activate the object.

*   none

    If the **sun.rmi.activation.execPolicy** property value is **none**, then the **rmid** command does not perform any validation of commands to start activation groups.

-log *dir*

Specifies the name of the directory the activation system daemon uses to write its database and associated information. The log directory defaults to creating a log, in the directory in which the **rmid** command was executed.

-port *port*

Specifies the port the registry uses. The activation system daemon binds the **ActivationSystem**, with the name **java.rmi.activation.ActivationSystem**, in this registry. The **ActivationSystem** on the local machine can be obtained using the following **Naming.lookup** method call:

import java.rmi.*;
  import java.rmi.activation.*;
  ActivationSystem system; system = (ActivationSystem)
  Naming.lookup("//:port/java.rmi.activation.ActivationSystem");

-stop
>   Stops the current invocation of the **rmid** command for a port specified by the **-port** option. If no
>   port is specified, then this option stops the **rmid** invocation running on port 1098.

ENVIRONMENT VARIABLES
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
- java(1)

- Setting the Class Path