

NAME

`jcmd` – send diagnostic command requests to a running Java Virtual Machine (JVM)

SYNOPSIS

`jcmd` [*pid* | *main-class*] *command*... | **PerfCounter.print** | **-f** *filename*

`jcmd` [-1]

`jcmd` -h

pid When used, the `jcmd` utility sends the diagnostic command request to the process ID for the Java process.

main-class

When used, the `jcmd` utility sends the diagnostic command request to all Java processes with the specified name of the main class.

command

The **command** must be a valid `jcmd` command for the selected JVM. The list of available commands for `jcmd` is obtained by running the **help** command (`jcmd pid help`) where *pid* is the process ID for the running Java process. If the *pid* is 0, commands will be sent to all Java processes. The main class argument will be used to match, either partially or fully, the class used to start Java. If no options are given, it lists the running Java process identifiers with the main class and command-line arguments that were used to launch the process (the same as using -1).

Perfcounter.print

Prints the performance counters exposed by the specified Java process.

-f *filename*

Reads and executes commands from a specified file, *filename*.

-1 Displays the list of Java Virtual Machine process identifiers that are not running in a separate docker process along with the main class and command-line arguments that were used to launch the process. If the JVM is in a docker process, you must use tools such as **ps** to look up the PID.

Note:

Using `jcmd` without arguments is the same as using `jcmd -1`.

-h Displays the `jcmd` utility's command-line help.

DESCRIPTION

The `jcmd` utility is used to send diagnostic command requests to the JVM. It must be used on the same machine on which the JVM is running, and have the same effective user and group identifiers that were used to launch the JVM. Each diagnostic command has its own set of arguments. To display the description, syntax, and a list of available arguments for a diagnostic command, use the name of the command as the argument. For example:

```
jcmd pid help command
```

If arguments contain spaces, then you must surround them with single or double quotation marks (' or "). In addition, you must escape single or double quotation marks with a backslash (\) to prevent the operating system shell from processing quotation marks. Alternatively, you can surround these arguments with single quotation marks and then with double quotation marks (or with double quotation marks and then with single quotation marks).

If you specify the process identifier (*pid*) or the main class (*main-class*) as the first argument, then the `jcmd` utility sends the diagnostic command request to the Java process with the specified identifier or to all Java processes with the specified name of the main class. You can also send the diagnostic command request to all available Java processes by specifying 0 as the process identifier.

COMMANDS FOR JCMD

The *command* must be a valid `jcmd` diagnostic command for the selected JVM. The list of available commands for `jcmd` is obtained by running the **help** command (`jcmd pid help`) where *pid* is the process ID for a running Java process. If the *pid* is 0, commands will be sent to all Java processes. The main class ar-

gument will be used to match, either partially or fully, the class used to start Java. If no options are given, it lists the running Java process identifiers that are not in separate docker processes along with the main class and command-line arguments that were used to launch the process (the same as using `-l`).

The following commands are available:

help [*options*] [*arguments*]

For more information about a specific command.

arguments:

- *command name*: The name of the command for which we want help (STRING, no default value)

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **-all**: (Optional) Show help for all commands (BOOLEAN, false) .

Compiler.codecache

Prints code cache layout and bounds.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

Compiler.codelist

Prints all compiled methods in code cache that are alive.

Impact: Medium

Permission: `java.lang.management.ManagementPermission(monitor)`

Compiler.queue

Prints methods queued for compilation.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

Compiler.directives_add *filename* *arguments*

Adds compiler directives from a file.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

arguments:

filename: The name of the directives file (STRING, no default value)

Compiler.directives_clear

Remove all compiler directives.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

Compiler.directives_print

Prints all active compiler directives.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

Compiler.directives_remove

Remove latest added compiler directive.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

GC.class_histogram [*options*]

Provides statistics about the Java heap usage.

Impact: High — depends on Java heap size and content.

Permission: `java.lang.management.ManagementPermission(monitor)`

Note:

The *options* must be specified using either *key* or *key=value* syntax.

options:

- **-all:** (Optional) Inspects all objects, including unreachable objects (BOOLEAN, false)

GC.finalizer_info

Provides information about the Java finalization queue.

Impact: Medium

Permission: `java.lang.management.ManagementPermission(monitor)`

GC.heap_dump [*options*] [*arguments*]

Generates a HPROF format dump of the Java heap.

Impact: High — depends on the Java heap size and content. Request a full GC unless the **-all** option is specified.

Permission: `java.lang.management.ManagementPermission(monitor)`

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **-all:** (Optional) Dump all objects, including unreachable objects (BOOLEAN, false)

arguments:

- *filename:* The name of the dump file (STRING, no default value)

GC.heap_info

Provides generic Java heap information.

Impact: Medium

Permission: `java.lang.management.ManagementPermission(monitor)`

GC.run

Calls `java.lang.System.gc()`.

Impact: Medium — depends on the Java heap size and content.

GC.run_finalization

Calls `java.lang.System.runFinalization()`.

Impact: Medium — depends on the Java content.

JFR.check [*options*]

Show information about a running flight recording

Impact: Low

Note:

The *options* must be specified using either *key* or *key=value* syntax. If no parameters are entered, information for all active recordings is shown.

options:

- **name:** (Optional) Name of the flight recording. (STRING, no default value)
- **verbose:** (Optional) Flag for printing the event settings for the recording (BOOLEAN, false)

JFR.configure [*options*]

Set the parameters for a flight recording

Impact: Low

Note:

The *options* must be specified using either *key* or *key=value* syntax. If no parameters are entered, the current settings are displayed.

options:

- **globalbuffercount:** (Optional) Number of global buffers. This option is a legacy option: change the **memorysize** parameter to alter the number of global buffers. This value cannot be changed once JFR has been initialized. (STRING, default determined by the value for **memorysize**)
- **globalbuffersize:** (Optional) Size of the global buffers, in bytes. This option is a legacy option: change the **memorysize** parameter to alter the size of the global buffers. This value cannot be changed once JFR has been initialized. (STRING, default determined by the value for **memorysize**)
- **maxchunksize:** (Optional) Maximum size of an individual data chunk in bytes if one of the following suffixes is not used: 'm' or 'M' for megabytes OR 'g' or 'G' for gigabytes. This value cannot be changed once JFR has been initialized. (STRING, 12M)
- **memorysize:** (Optional) Overall memory size, in bytes if one of the following suffixes is not used: 'm' or 'M' for megabytes OR 'g' or 'G' for gigabytes. This value cannot be changed once JFR has been initialized. (STRING, 10M)
- **repositorypath:** (Optional) Path to the location where recordings are stored until they are written to a permanent file. (STRING, The default location is the temporary directory for the operating system. On Linux operating systems, the temporary directory is **/tmp**. On Windows, the temporary directory is specified by the **TMP** environment variable.)
- **stackdepth:** (Optional) Stack depth for stack traces. Setting this value greater than the default of 64 may cause a performance degradation. This value cannot be changed once JFR has been initialized. (LONG, 64)
- **thread_buffer_size:** (Optional) Local buffer size for each thread in bytes if one of the following suffixes is not used: 'k' or 'K' for kilobytes or 'm' or 'M' for megabytes. Overriding this parameter could reduce performance and is not recommended. This value cannot be changed once JFR has been initialized. (STRING, 8k)
- **samplethreads:** (Optional) Flag for activating thread sampling. (BOOLEAN, true)

JFR.dump [*options*]

Write data to a file while a flight recording is running

Impact: Low

Note:

The *options* must be specified using either *key* or *key=value* syntax. No options are required. The recording continues to run after the data is written.

options:

- **begin:** (Optional) Specify the time from which recording data will be included in the dump file. The format is specified as local time. (STRING, no default value)
- **end:** (Optional) Specify the time to which recording data will be included in the dump file. The format is specified as local time. (STRING, no default value)

Note: For both **begin** and **end**, the time must be in a format that can be read by `java.time.LocalDateTime::parse(String)`, `java.time.LocalDateTime::parse(String)` or `java.time.Instant::parse(String)`. For example, "13:20:15", "2020-03-17T09:00:00" or "2020-03-17T09:00:00Z".

Note: **begin** and **end** times correspond to the timestamps found within the recorded information in the flight recording data.

Another option is to use a time relative to the current time that is specified by a negative integer followed by "s", "m" or "h". For example, "-12h", "-15m" or "-30s"

- **filename:** (Optional) Name of the file to which the flight recording data is dumped. If no filename is given, a filename is generated from the PID and the current date. The filename may also be a directory in which case, the filename is generated from the PID and the current date in the specified directory. (STRING, no default value)
- **maxage:** (Optional) Length of time for dumping the flight recording data to a file. (INTEGER followed by 's' for seconds 'm' for minutes or 'h' for hours, no default value)
- **maxsize:** (Optional) Maximum size for the amount of data to dump from a flight recording in bytes if one of the following suffixes is not used: 'm' or 'M' for megabytes OR 'g' or 'G' for gigabytes. (STRING, no default value)
- **name:** (Optional) Name of the recording. If no name is given, data from all recordings is dumped. (STRING, no default value)
- **path-to-gc-root:** (Optional) Flag for saving the path to garbage collection (GC) roots at the time the recording data is dumped. The path information is useful for finding memory leaks but collecting it can cause the application to pause for a short period of time. Turn on this flag only when you have an application that you suspect has a memory leak. (BOOLEAN, false)

JFR.start [*options*]

Start a flight recording

Impact: Low

Note:

The *options* must be specified using either *key* or *key=value* syntax. If no parameters are entered, then a recording is started with default values.

options:

- **delay:** (Optional) Length of time to wait before starting to record (INTEGER followed by 's' for seconds 'm' for minutes or 'h' for hours, 0s)
- **disk:** (Optional) Flag for also writing the data to disk while recording (BOOLEAN, true)
- **dumponexit:** (Optional) Flag for writing the recording to disk when the Java Virtual Machine (JVM) shuts down. If set to 'true' and no value is given for **filename**, the recording is written to a file in the directory where the process was started. The file name is a system-generated name that contains the process ID, the recording ID and the current time stamp. (For example: **id-1-2019_12_12_10_41.jfr**) (BOOLEAN, false)
- **duration:** (Optional) Length of time to record. Note that **0s** means forever (INTEGER followed by 's' for seconds 'm' for minutes or 'h' for hours, 0s)
- **filename:** (Optional) Name of the file to which the flight recording data is written when the recording is stopped. If no filename is given, a filename is generated from the PID and the current date and is placed in the directory where the process was started. The filename may also be a directory in which case, the filename is generated from the PID and the current date in the specified directory. (STRING, no default value)
- **maxage:** (Optional) Maximum time to keep the recorded data on disk. This parameter is valid only when the **disk** parameter is set to **true**. Note **0s** means forever. (INTEGER followed

by 's' for seconds 'm' for minutes or 'h' for hours, 0s)

- **maxsize:** (Optional) Maximum size of the data to keep on disk in bytes if one of the following suffixes is not used: 'm' or 'M' for megabytes OR 'g' or 'G' for gigabytes. This parameter is valid only when the **disk** parameter is set to 'true'. The value must not be less than the value for the **maxchunksize** parameter set with the **JFR.configure** command. (STRING, 0 (no maximum size))
- **name:** (Optional) Name of the recording. If no name is provided, a name is generated. Make note of the generated name that is shown in the response to the command so that you can use it with other commands. (STRING, system-generated default name)
- **path-to-gc-root:** (Optional) Flag for saving the path to garbage collection (GC) roots at the end of a recording. The path information is useful for finding memory leaks but collecting it is time consuming. Turn on this flag only when you have an application that you suspect has a memory leak. If the **settings** parameter is set to 'profile', then the information collected includes the stack trace from where the potential leaking object was allocated. (BOOLEAN, false)
- **settings:** (Optional) Name of the settings file that identifies which events to record. To specify more than one file, separate the names with a comma (','),. Include the path if the file is not in **JAVA-HOME/lib/jfr**. The following profiles are included with the JDK in the **JAVA-HOME/lib/jfr** directory: 'default.jfc': collects a predefined set of information with low overhead, so it has minimal impact on performance and can be used with recordings that run continuously; 'profile.jfc': Provides more data than the 'default.jfc' profile, but with more overhead and impact on performance. Use this configuration for short periods of time when more information is needed. Use **none** to start a recording without a predefined configuration file. (STRING, **JAVA-HOME/lib/jfr/default.jfc**)

Event settings and .jfc options can be specified using the following syntax:

- **option:** (Optional) Specifies the option value to modify. To list available options, use the **JAVA-HOME/bin/jfr** tool.
- **event-setting:** (Optional) Specifies the event setting value to modify. Use the form: **#=** To add a new event setting, prefix the event name with '+'.

You can specify values for multiple event settings and .jfc options by separating them with a whitespace. In case of a conflict between a parameter and a .jfc option, the parameter will take precedence. The whitespace delimiter can be omitted for timespan values, i.e. 20ms. For more information about the settings syntax, see Javadoc of the **jdk.jfr** package.

JFR.stop [*options*]

Stop a flight recording

Impact: Low

Note:

The *options* must be specified using either *key* or *key=value* syntax. If no parameters are entered, then no recording is stopped.

options:

- **filename:** (Optional) Name of the file to which the recording is written when the recording is stopped. If no path is provided, the data from the recording is discarded. (STRING, no default value)
- **name:** (Optional) Name of the recording (STRING, no default value)

JVMTI.agent_load [*arguments*]

Loads JVMTI native agent.

Impact: Low

Permission: `java.lang.management.ManagementPermission(control)`

arguments:

- *library path*: Absolute path of the JVMTI agent to load. (STRING, no default value)
- *agent option*: (Optional) Option string to pass the agent. (STRING, no default value)

JVMTI.data_dump

Signals the JVM to do a data-dump request for JVMTI.

Impact: High

Permission: `java.lang.management.ManagementPermission(monitor)`

ManagementAgent.start [*options*]

Starts remote management agent.

Impact: Low — no impact

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **config.file**: (Optional) Sets `com.sun.management.config.file` (STRING, no default value)
- **jmxremote.host**: (Optional) Sets `com.sun.management.jmxremote.host` (STRING, no default value)
- **jmxremote.port**: (Optional) Sets `com.sun.management.jmxremote.port` (STRING, no default value)
- **jmxremote.rmi.port**: (Optional) Sets `com.sun.management.jmxremote.rmi.port` (STRING, no default value)
- **jmxremote.ssl**: (Optional) Sets `com.sun.management.jmxremote.ssl` (STRING, no default value)
- **jmxremote.registry.ssl**: (Optional) Sets `com.sun.management.jmxremote.registry.ssl` (STRING, no default value)
- **jmxremote.authenticate**: (Optional) Sets `com.sun.management.jmxremote.authenticate` (STRING, no default value)
- **jmxremote.password.file**: (Optional) Sets `com.sun.management.jmxremote.password.file` (STRING, no default value)
- **jmxremote.access.file**: (Optional) Sets `com.sun.management.jmxremote.access.file` (STRING, no default value)
- **jmxremote.login.config**: (Optional) Sets `com.sun.management.jmxremote.login.config` (STRING, no default value)
- **jmxremote.ssl.enabled.cipher.suites**: (Optional) Sets `com.sun.management.jmxremote.ssl.enabled.cipher.suites` (STRING, no default value)
- **jmxremote.ssl.enabled.cipher.suite**: (Optional) Sets `com.sun.management.jmxremote.ssl.enabled.cipher.suite` (STRING, no default value)
- **jmxremote.ssl.enabled.protocols**: (Optional) Sets `com.sun.management.jmxremote.ssl.enabled.protocols` (STRING, no default value)
- **jmxremote.ssl.need.client.auth**: (Optional) Sets `com.sun.management.jmxremote.ssl.need.client.auth` (STRING, no default value)
- **jmxremote.ssl.config.file**: (Optional) Sets `com.sun.management.jmxremote.ssl.config.file` (STRING, no default value)

- **jmxremote.autodiscovery:** (Optional) Sets `com.sun.management.jmxremote.au todiscovery` (STRING, no default value)
- **jdp.port:** (Optional) Sets `com.sun.management.jdp.port` (INT, no default value)
- **jdp.address:** (Optional) Sets `com.sun.management.jdp.address` (STRING, no default value)
- **jdp.source_addr:** (Optional) Sets `com.sun.management.jdp.source_addr` (STRING, no default value)
- **jdp.ttl:** (Optional) Sets `com.sun.management.jdp.ttl` (INT, no default value)
- **jdp.pause:** (Optional) Sets `com.sun.management.jdp.pause` (INT, no default value)
- **jdp.name:** (Optional) Sets `com.sun.management.jdp.name` (STRING, no default value)

ManagementAgent.start_local

Starts the local management agent.

Impact: Low — no impact

ManagementAgent.status

Print the management agent status.

Impact: Low — no impact

Permission: `java.lang.management.ManagementPermission(monitor)`

ManagementAgent.stop

Stops the remote management agent.

Impact: Low — no impact

Thread.print [*options*]

Prints all threads with stacktraces.

Impact: Medium — depends on the number of threads.

Permission: `java.lang.management.ManagementPermission(monitor)`

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **-l:** (Optional) Prints `java.util.concurrent` locks (BOOLEAN, false)

VM.cds [*arguments*]

Dumps a static or dynamic shared archive that includes all currently loaded classes.

Impact: Medium — pause time depends on number of loaded classes

Permission: `java.lang.management.ManagementPermission(monitor)`

arguments:

- **subcmd:** must be either `static_dump` or
- **filename:** (Optional) Name of the shared archive to be dumped (STRING, no default value)

If **filename** is not specified, a default file name is chosen using the pid of the target JVM process. For example, `java_pid1234_static.jsa`, `java_pid5678_dynamic.jsa`, etc.

If **filename** is not specified as an absolute path, the archive file is created in a directory relative to the current directory of the target JVM process.

VM.classloaders [*options*]

Prints classloader hierarchy.

Impact: Medium ——— Depends on number of class loaders and classes loaded.

Permission: `java.lang.management.ManagementPermission(monitor)`

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **show-classes**: (Optional) Print loaded classes. (BOOLEAN, false)
- **verbose**: (Optional) Print detailed information. (BOOLEAN, false)
- **fold**: (Optional) Show loaders of the same name and class as one. (BOOLEAN, true)

VM.classloader_stats

Prints statistics about all ClassLoaders.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

VM.class_hierarchy [*options*] [*arguments*]

Prints a list of all loaded classes, indented to show the class hierarchy. The name of each class is followed by the `ClassLoaderData*` of its `ClassLoader`, or "null" if it is loaded by the bootstrap class loader.

Impact: Medium ——— depends on the number of loaded classes.

Permission: `java.lang.management.ManagementPermission(monitor)`

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **-i**: (Optional) Inherited interfaces should be printed. (BOOLEAN, false)
- **-s**: (Optional) If a class name is specified, it prints the subclasses. If the class name is not specified, only the superclasses are printed. (BOOLEAN, false)

arguments:

- **classname**: (Optional) The name of the class whose hierarchy should be printed. If not specified, all class hierarchies are printed. (STRING, no default value)

VM.command_line

Prints the command line used to start this VM instance.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

VM.dynlibs

Prints the loaded dynamic libraries.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

VM.info

Prints information about the JVM environment and status.

Impact: Low

Permission: `java.lang.management.ManagementPermission(monitor)`

VM.log [*options*]

Lists current log configuration, enables/disables/configures a log output, or rotates all logs.

Impact: Low

Permission: `java.lang.management.ManagementPermission(control)`

options:

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

- **output:** (Optional) The name or index (#) of output to configure. (STRING, no default value)
- **output_options:** (Optional) Options for the output. (STRING, no default value)
- **what:** (Optional) Configures what tags to log. (STRING, no default value)
- **decorators:** (Optional) Configures which decorators to use. Use 'none' or an empty value to remove all. (STRING, no default value)
- **disable:** (Optional) Turns off all logging and clears the log configuration. (BOOLEAN, no default value)
- **list:** (Optional) Lists current log configuration. (BOOLEAN, no default value)
- **rotate:** (Optional) Rotates all logs. (BOOLEAN, no default value)

VM.flags [*options*]

Prints the VM flag options and their current values.

Impact: Low

Permission: **java.lang.management.ManagementPermission(monitor)**

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **-all:** (Optional) Prints all flags supported by the VM (BOOLEAN, false).

VM.native_memory [*options*]

Prints native memory usage

Impact: Medium

Permission: **java.lang.management.ManagementPermission(monitor)**

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **summary:** (Optional) Requests runtime to report current memory summary, which includes total reserved and committed memory, along with memory usage summary by each subsystem. (BOOLEAN, false)
- **detail:** (Optional) Requests runtime to report memory allocation >= 1K by each callsite. (BOOLEAN, false)
- **baseline:** (Optional) Requests runtime to baseline current memory usage, so it can be compared against in later time. (BOOLEAN, false)
- **summary.diff:** (Optional) Requests runtime to report memory summary comparison against previous baseline. (BOOLEAN, false)
- **detail.diff:** (Optional) Requests runtime to report memory detail comparison against previous baseline, which shows the memory allocation activities at different callsites. (BOOLEAN, false)
- **shutdown:** (Optional) Requests runtime to shutdown itself and free the memory used by runtime. (BOOLEAN, false)
- **statistics:** (Optional) Prints tracker statistics for tuning purpose. (BOOLEAN, false)

- **scale:** (Optional) Memory usage in which scale, KB, MB or GB (STRING, KB)

VM.print_touched_methods

Prints all methods that have ever been touched during the lifetime of this JVM.

Impact: Medium ——— depends on Java content.

VM.set_flag [*arguments*]

Sets the VM flag option by using the provided value.

Impact: Low

Permission: `java.lang.management.ManagementPermission(control)`

arguments:

- *flag name:* The name of the flag that you want to set (STRING, no default value)
- *string value:* (Optional) The value that you want to set (STRING, no default value)

VM.stringtable [*options*]

Dumps the string table.

Impact: Medium ——— depends on the Java content.

Permission: `java.lang.management.ManagementPermission(monitor)`

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **-verbose:** (Optional) Dumps the content of each string in the table (BOOLEAN, false)

VM.symboltable [*options*]

Dumps the symbol table.

Impact: Medium ——— depends on the Java content.

Permission: `java.lang.management.ManagementPermission(monitor)`

Note:

The following *options* must be specified using either *key* or *key=value* syntax).

options:

- **-verbose:** (Optional) Dumps the content of each symbol in the table (BOOLEAN, false)

VM.systemdictionary

Prints the statistics for dictionary hashtable sizes and bucket length.

Impact: Medium

Permission: `java.lang.management.ManagementPermission(monitor)`

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **verbose:** (Optional) Dump the content of each dictionary entry for all class loaders (BOOLEAN, false) .

VM.system_properties

Prints the system properties.

Impact: Low

Permission: `java.util.PropertyPermission(*, read)`

VM.uptime [*options*]

Prints the VM uptime.

Impact: Low

Note:

The following *options* must be specified using either *key* or *key=value* syntax.

options:

- **-date**: (Optional) Adds a prefix with the current date (BOOLEAN, false)

VM.version

Prints JVM version information.

Impact: Low

Permission: `java.util.PropertyPermission(java.vm.version, read)`