Name generate-jvm-report– shows the threads, classes, memory, and loggers for a given target instance.

Synopsis

```
generate-jvm-report
[--help]
[ --type =summary|memory|class|thread|log] [target]
```

Description The `generate-jvm-report` subcommand creates a report that shows the threads (dump of stack trace), classes, memory, or loggers for a given target instance, including the domain administration server (DAS). This subcommand works only with the Enterprise Server instance processes. This subcommand provides an alternative to sending Ctrl+Break or `kill -3` signals to Enterprise Server processes to obtain a stack trace for processes that are hanging.

The information in the report is obtained from managed beans (MBeans) and MXBeans that are provided in the Java™ Platform, Standard Edition (Java SE) or JDK™ software with which Enterprise Server is being used.

If Enterprise Server is running in the Java Runtime Environment (JRE™) software from JDK release 6 or Java SE 6, additional information is provided, for example:

- System load on the available processors
- Object monitors that are currently held or requested by a thread
- Lock objects that a thread is holding, for example, `ReentrantLock` objects and `ReentrantReadWriteLock` objects

If the JRE software cannot provide this information, the report contains the text `NOT_AVAILABLE`.

This subcommand is supported in remote mode only. The subcommand does not work if the target server instance is not running.

Options

`--help`
Displays the help text for the subcommand.

`--type`
The type of report that is to be generated.

`summary`
Displays summary information about the threads, classes, and memory (default).

`memory`
Provides information about heap and non-heap memory consumption, memory pools, and garbage collection statistics for a given target instance.

`class`
Provides information about the class loader for a given target instance.

`thread`
Provides information about threads running and the thread dump (stack trace) for a given target instance.

`log`
Provides information about the loggers that are registered in the Virtual Machine for the Java platform (Java Virtual Machine or JVM™ machine).[1]

Operands *target*
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

Examples **EXAMPLE 1** Obtaining Summary Information About Threads, Classes, and Memory

This example generates a report of the type `summary` for `server1`.

```
asadmin> generate-jvm-report --type summary server1
Operating System Information:
Name of the Operating System: Linux
Binary Architecture name of the Operating System: i386, Version:
2.6.9-22.ELsmp
Number of processors available on the Operating System: 2
...
user.language = en
user.name = root
user.timezone = America/Los_Angeles
Command generate-jvm-report executed successfully
```

EXAMPLE 2 Obtaining Information About Memory Usage

This example generates a report of the type `memory`.

```
asadmin> generate-jvm-report --type=memory
Memory Pool Name: Eden Space
Memory that Java Virtual Machine initially requested to the
Operating System: 2,097,152 Bytes
Memory that Java Virtual Machine is guaranteed to receive from the
Operating System: 9,895,936 Bytes
Maximum Memory that Java Virtual Machine may get from the
Operating System: 168,427,520 Bytes
Note that this is not guaranteed.
Memory that Java Virtual Machine uses at this time: 7,159,784 Bytes
...
Memory Pool Name: Survivor Space
Memory that Java Virtual Machine initially requested to the
Operating System: 65,536 Bytes
Memory that Java Virtual Machine is guaranteed to receive from the
Operating System: 262,144 Bytes
Maximum Memory that Java Virtual Machine may get from the
```

[1] The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.

EXAMPLE 2 Obtaining Information About Memory Usage *(Continued)*

```
Operating System: 5,242,880 Bytes.
Note that this is not guaranteed.
Memory that Java Virtual Machine uses at this time: 35,208 Bytes
...
Name of the Garbage Collector: MarkSweepCompact
Number of collections occured using this garbage collector: 0 Bytes
Garbage Collection Time: 0 Seconds 0 Milliseconds
Name of the Garbage Collector: Copy
Number of collections occured using this garbage collector: 47 Bytes
Garbage Collection Time: 1 Seconds 395 Milliseconds

Heap Memory Usage:
Memory that Java Virtual Machine initially requested to the
Operating System: 0 Bytes
Memory that Java Virtual Machine is guaranteed to receive from the
Operating System: 30,728,192 Bytes
Maximum Memory that Java Virtual Machine may get from the
Operating System: 531,628,032 Bytes.
Note that this is not guaranteed.
Memory that Java Virtual Machine uses at this time: 25,434,432 Bytes

Non-heap Memory Usage:
Memory that Java Virtual Machine initially requested to the
Operating System: 29,523,968 Bytes
Memory that Java Virtual Machine is guaranteed to receive from the
Operating System: 32,833,536 Bytes
Maximum Memory that Java Virtual Machine may get from the
Operating System: 121,634,816 Bytes
 Note that this is not guaranteed.
Memory that Java Virtual Machine uses at this time: 22,920,624 Bytes

Approximate number of objects for which finalization is pending: 0
Command generate-jvm-report executed successfully.
```

EXAMPLE 3 Obtaining Information About Running Threads

This example generates a report of the type `thread`.

```
asadmin> generate-jvm-report --type=thread
Full Java Thread Dump Java HotSpot(TM) Client VM 1.5.0_14-b03 Sun Microsystems Inc.
Number of threads: 39
Number of daemon threads: 33
Peak live thread count since the Java virtual machine started or peak was reset: 44
Is support for thread contention monitoring available on this JVM? [true]
Is thread contention monitoring enabled? [false]. If false, some thread
synchronization statistics are not be available.
```

EXAMPLE 3 Obtaining Information About Running Threads *(Continued)*

```
Is support for CPU time measurement for any thread available on this JVM? [true]
Is thread CPU time measurement enabled? [true]. If false, thread execution times
are not available for any thread.
------------------------------------------------------------------------------
Thread Execution Information:
----------------------
Thread "RMI ConnectionExpiration-[129.146.11.147:61218]" thread-id: 84 thread-state:
TIMED_WAITING
     at: java.lang.Thread.sleep(Native Method)
     at: sun.rmi.transport.tcp.TCPChannel$Reaper.run(TCPChannel.java:446)
     at: java.lang.Thread.run(Thread.java:595)
Thread Synchronization Statistics:
----------------------
Number of times this thread was blocked (to enter/reenter a Monitor): 0
Number of times this thread waited for a notification (i.e. it was in WAITING or
TIMED_WAITING state): 0
Total CPU time for this thread: 0 seconds 131,855 nanoseconds.
User-level CPU time for this thread: 0 seconds 131,855 nanoseconds.
Object Monitors currently held or requested by this thread: NOT_AVAILABLE
Ownable Synchronizers (e.g. ReentrantLock and ReentrantReadWriteLock) held by
this thread: NOT_AVAILABLE
------------------------------------------------------------------------------
...
Thread Execution Information:
----------------------
Thread "Reference Handler" thread-id: 2 thread-state: WAITING Waiting on lock:
java.lang.ref.Reference$Lock@f63055
     at: java.lang.Object.wait(Native Method)
     at: java.lang.Object.wait(Object.java:474)
     at: java.lang.ref.Reference$ReferenceHandler.run(Reference.java:116)
Thread Synchronization Statistics:
----------------------
Number of times this thread was blocked (to enter/reenter a Monitor): 318
Number of times this thread waited for a notification (i.e. it was in WAITING or
TIMED_WAITING state): 43
Total CPU time for this thread: 0 seconds 26,004,119 nanoseconds.
User-level CPU time for this thread: 0 seconds 26,004,119 nanoseconds.
Object Monitors currently held or requested by this thread: NOT_AVAILABLE
Ownable Synchronizers (e.g. ReentrantLock and ReentrantReadWriteLock) held by this
thread: NOT_AVAILABLE
No deadlock found

Command generate-jvm-report executed successfully.
```

EXAMPLE 4 Obtaining Information About a Class Loader

This example generates a report of the type `class`.

```
asadmin> generate-jvm-report --type=class
Class loading and unloading in the Java Virtual Machine:
Number of classes currently loaded in the Java Virtual Machine: 2,798
Number of classes loaded in the Java Virtual Machine since the startup: 2,798
Number of classes unloaded from the Java Virtual Machine: 0
Just-in-time (JIT) compilation information in the Java Virtual Machine:
Java Virtual Machine compilation monitoring allowed: true
Name of the Just-in-time (JIT) compiler: HotSpot Client Compiler
Total time spent in compilation: 0 Hours 0 Minutes 2 Seconds

Command generate-jvm-report executed successfully.
```

EXAMPLE 5 Obtaining Information About Loggers

This example generates a report for the type `log`.

```
asadmin> generate-jvm-report --type=log
Effective logging properties file:
[/home/someuser/glassfishv3-prelude/glassfish/domains/domain1/config/
logging.properties].
If null, it indicates JRE standard file.
Number of loggers currently registered in the JVM: [35]. Details follow:
If the level is blank, it is inherited from parent logger
Parent logger is the nearest existing parent logger
Logger Name | Logging Level | Parent Logger Name
-------------------------------------------------
|INFO|root
-------------------------------------------------
GRIZZLY||root
-------------------------------------------------
Grizzly||root
-------------------------------------------------
com.sun.enterprise.naming||root
-------------------------------------------------
com.sun.enterprise.v3.admin.commands||root
-------------------------------------------------
com.sun.enterprise.v3.server.DynamicReloadService||root
-------------------------------------------------
com.sun.enterprise.v3.server.DynamicReloader||root
-------------------------------------------------
com.sun.grizzly.util.http.HttpRequestURIDecoder||root
-------------------------------------------------
com.sun.jmx.remote.opt.util||root
-------------------------------------------------
```

EXAMPLE 5 Obtaining Information About Loggers *(Continued)*

```
global||root
------------------------------------------------
grizzly||root
------------------------------------------------
javax.enterprise.resource.jta||root
------------------------------------------------
javax.enterprise.resource.resourceadapter||root
------------------------------------------------
...
sun.rmi.transport.misc||root
------------------------------------------------
sun.rmi.transport.tcp||root
------------------------------------------------
sun.rmi.transport.tcp.proxy||sun.rmi.transport.tcp
------------------------------------------------
Command generate-jvm-report executed successfully.
```

Exit Status

0	subcommand executed successfully
1	error in executing the subcommand

See Also `create-jvm-options`(1), `delete-jvm-options`(1), `list-jvm-options(1)`

`asadmin(1M)`