

# MBeanCmd

*User's guide*

*v2.2*

---

# MBeanCmd: User's guide

JASMINe Team

Publication date \$Id: mbeancmd\_guide.xml 8156 2011-05-13 15:40:30Z jlegrand \$

Copyright © 2008-2010 Bull SAS



This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/deed.en> [<http://creativecommons.org/licenses/by/2.0/>] or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

---

# Table of Contents

1. Introduction .....	1
2. MBeans and your J2EE server .....	2
2.1. What is an MBean ? .....	2
2.2. MBeans in JOnAS .....	2
2.3. Connecting to MBeans on a J2EE server .....	2
3. Using MBeanCmd .....	4
3.1. Getting help .....	4
3.2. Defining targets .....	4
3.3. Defining outputs .....	5
3.4. The Poll command .....	5
3.5. The Stat command .....	7
3.6. The JDBC Connections command .....	8
3.7. The Dump command .....	8
3.8. The Snap command .....	9
3.9. The Scan command .....	11
3.10. The mbean command .....	12
3.11. The WildStat command .....	13
3.12. Example: output the number of threads .....	14
3.12.1. On the console .....	15
3.12.2. On a spreadsheet .....	15
3.12.3. In a graph .....	15
4. MGen - a configuration generator for MBeanCmd .....	17
4.1. Pre-Requisites .....	17
4.2. MGen setup .....	17
4.2.1. Tree structure .....	17
4.2.2. Compile MGen .....	18
4.3. mgen-conf.xml ... scheme summary of THE file .....	18
4.3.1. "vars" section .....	18
4.3.2. "components" section .....	19
4.3.3. "servers" section .....	19

---

## List of Tables

3.1. Polled MBeans and obtained indicators .....	6
--	---

---

## List of Examples

3.1. -target option use .....	5
3.2. Result format for poll -tx .....	7
3.3. Poll command examples .....	7
3.4. JDBC Connection status .....	8
3.5. The Snap -s option .....	10
3.6. The Snap -d option .....	11
3.7. The Scan command .....	12
3.8. The Scan command : -delta option .....	12
3.9. MBean command examples .....	13
3.10. The WildStat command examples .....	14

---

# Chapter 1. Introduction

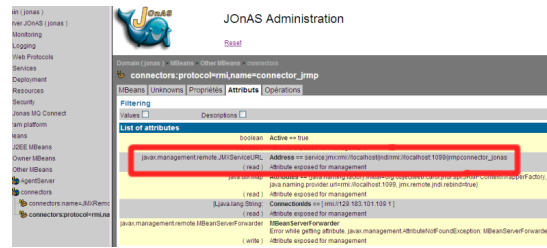
MBeanCmd is a command-line tool written in Java for interacting with MBeans on target J2EE servers.

It can :

- Send commands to :
  - Any number of MBeans
  - On any number of target J2EE servers
  - In a periodical way
- Receive the result of that command, for example the CPU usage rate measured by the JVM MBean
- Use this result to :
  - Display it directly on the console
  - Output it to a text file
  - Display it on a graph
  - Push it to a JASMINe Event Switch

It does not depend on any J2EE package. The various outputs (for example, file, graph and JASMINe Event Switch) can be combined.





The screenshot displays the JOnAS Administration web interface. On the left is a navigation tree with categories like Monitoring, Logging, Web Protocols, Services, Deployment, Resources, Security, JMS, J2EE, and JMS. The main panel shows the configuration for the MBean `connectors:protocolrmi:connector_jmp`. The `Attributes` tab is selected, showing a list of attributes. One attribute, `java.management.remote.j2ee.ServiceURL`, is highlighted with a red box. Its value is `Address = service:jmx:rmi://localhost:1099/jmpconnector_jmp`. Below the list, there is a section for `Operations` with methods like `read`, `write`, and `destroy`.

connectors:protocolrmi:connector\_jmp

Attributes

java.management.remote.j2ee.ServiceURL Address = service:jmx:rmi://localhost:1099/jmpconnector\_jmp

Operations

read write destroy



# Chapter 3. Using MBeanCmd

## 3.1. Getting help

To launch MBeanCmd, just go to the folder where mbean.jar is present and launch :

```
java -jar mbean.jar
```

Since no option has been specified, this will simply display the help with the list of available commands.

To display the help for a command, say **poll**, type in:

```
java -jar mbean.jar help poll
```



### Note

Note that the help file for poll is quite large: it includes the usage options and the DTD to use to display a graph.

## 3.2. Defining targets

To use MBeanCmd for capturing statistics periodically (**stat** and **poll** commands) or to get a thread dump from a JOnAS server (**dump** command), you first need to define which server(s) you're targeting. This can be done in two ways:

- If you're targeting only one server, define **jasmine.jmx.url** system property with value set to the target's JMX connector URL.

We recommend to avoid this way if the result is pushed into JASMINe Monitoring.

- If you're targeting multiple servers, create a file called **probe-config.xml** in the current folder. This file contains a target entry for each server. A target entry contains the following attributes:

- **url**, JMX url address of a connector for the target.
- **id**, the target's name.
- **user**, the user name to access the target using a secured jmx connector.
- **password**, the password to access the target using a secured jmx connector.

For example:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<probe-config xmlns="org.ow2.jasmine.monitoring.mbeancmd:probe-config">
  <target url="service:jmx:rmi:///jndi/rmi://localhost:1199/
jrmpconnector_jasmine-monitoring" id="myself"/>
  <target url="service:jmx:rmi:///jndi/rmi://localhost:1099/
jrmpconnector_jasmine" id="jonas"/>
  <target url="service:jmx:rmi://localhost:37659/jndi/rmi://localhost:2063/
jrmpconnector_master" id="master" user="monitor" password="jonas"/>
</probe-config>
```

### Example 3.1. -target option use

```
poll ... -target master myself
```

```
poll ... -target all
```

The "all" target stands for all the targets defined by the configuration file plus the `jasmine.jmx.url` property.



#### Note

For several commands (as **poll -cpusun**), the targeted servers need to be launched with the system property **com.sun.management.jmxremote**, in order to enable the monitoring of this JVM by JMX.

## 3.3. Defining outputs

An output is a destination for the statistics obtained by **stat** and **poll** commands.

The following output types may be used in a command:

- **-console** : Causes the output to be redirected to standard output.
- **-f FILE** : Causes the output to be redirected to FILE.
- **-graph XML\_DEF** : Displays graphs according to the XML definitions specified by the XML\_DEF file. Output is also printed on the console.
- **-jasmine URL** : Causes the output to be redirected toward JASMINe Monitoring.

The URL has the following format: **tcp://host:port/MBeanCmd**.

where *port* is the value of the MBeanCmd port defined by the JASMINe Monitoring configuration.

The default value is *18564*.

All the options may be combined. Default is "-console".

## 3.4. The Poll command

Command allowing to periodically poll target servers and generate statistic indicators on several categories of resources of a Java EE Server.

**poll [-p period] [-s separator] bean [OUTPUT] [TARGET]**

**poll [-dtd]**

- **-p period** : Sets the poll period in seconds. Default value is 10s.
- **-s separator** : Sets the separator used to delimit the obtained monitoring data. Default is ";"
- **bean** : Allows to focus the polling on a category of indicators. It must be one of the followings
  - **-server** : Prints overall statistics.

- -http : Prints statistics on HTTP/AJP connectors (one line per connector).
- -tx : Prints statistics on transactions.
- -cpusun : Prints statistics on the CPU usage.
- -ds FILTER : Prints statistics on datasources.
- -jcacf FILTER : Prints statistics on JCA connection factories.
- -slb FILTER : Prints statistics on stateless session EJBs.
- -sfb FILTER : Prints statistics on stateful session EJBs.
- -ent FILTER : Prints statistics on entity beans.
- -servlet FILTER : Prints statistics on servlets.
- -joramq FILTER : Prints statistics on JORAM queues. The domain in FILTER shall be joramClient.
- TARGET  
See Section 3.2, "Defining targets"
- OUTPUT  
See Section 3.3, "Defining outputs"
- -dtd : prints out the XML DTD for the graphs.

You can find in the table bellow the list of indicators provided for each poll option along with the name of the used MBeans.

**Table 3.1. Polled MBeans and obtained indicators**

Option	MBeans and resulted indicators
-tx	<b>*:j2eeType=JTAResource,*</b> => pending (for current transactions), commit, rollback, timedOut
	<b>processed</b> => txRate, commitRatio, currentCommitRatio
-http	<b>*:type=Manager,*</b> => sessions (for active sessions), sessionCount
	<b>*:type=GlobalRequestProcessor,*</b> => requests, errors, maxTime, bytesSent, bytesReceived
	<b>*:type=ThreadPool,*</b> => pending (for current number of busy threads)
	<b>processed</b> => rate (for http requests rate), meanTime (of processing time per request), bytesReveivedRate, byteSentRate
-ds	<b>*:j2eeType=JDBCDataSource,*</b> => servedOpen, jdbcMaxConnPool, currentOpen, busyMax, waitersHighRecent, waitingHighRecent, rejectedOpen, connectionLeaks
	<b>processed</b> => servedRate, meanWaitTime
-ent	<b>*:j2eeType=EntityBean,*</b> => cacheSize, poolSize, maxCacheSize, minPoolSize, usedInTx, usedOutTx, unusedReady, markedRemoved, pkNumber, passivated
-slb	<b>*:j2eeType=StatelessSessionBean,*</b> => cacheSize, poolSize, minPoolSize, maxCacheSize, sessionTimeOut, totalProcessingTime, numberOfCalls

Option	MBeans and resulted indicators
	<b>processed</b> => callsRate, totalProcessingTimeReq
-sfb	<b>*:j2eeType=StatefulSessionBean,*</b> => cacheSize, poolSize, minPoolSize, maxCacheSize, sessionTimeout, totalProcessingTime, numberOfCalls <b>processed</b> => callsRate, totalProcessingTimeReq
-servlet	<b>*:j2eeType=Servlet,*</b> => requestCount, errorCount, maxTime, minTime <b>processed</b> => requestRate, errorRate, procTimeReq
-cpusun	<b>java.lang:type=OperatingSystem</b> => processCpuTime <b>processed</b> => currentCpuTime, currentCpuLoad

### Example 3.2. Result format for poll -tx

```
time;date;sname;server;domain;mbean;cmdid;txRate;pending;commit;rollback;timedOut;commitRatio;currentCommitRatio;
1292575217080;2010/12/17
09:40:17;target_jmx_url;serverName;domainName;jonas:J2EEServer=serverName,j2eeType=JTAResource,name=JTAResource
```

Where target\_jmx\_url has the following format service:jmx:rmi:///jndi/rmi://host:port/jrmpconnector\_serverName. In case of a JOnAS server, the serverName and the domainName are configuration parameters defined in jonas.properties file or using -n option for the serverName and using -Ddomain.name.

### Example 3.3. Poll command examples

```
poll -ds "*:j2eeType=JDBCDataSource,*"
poll -slb "*:j2eeType=StatelessSessionBean,J2EEApplication=myApp,*"
poll -sfb "*:j2eeType=StatefulSessionBean,J2EEApplication=myApp,*"
poll -ent "*:j2eeType=EntityBean,J2EEApplication=myApp,*"
poll -servlet "*:j2eeType=Servlet,J2EEApplication=myApp,*"
poll -joramq "joramclient:type=Queue,*"
```

## 3.5. The Stat command

Command allowing to periodically poll MBeans on target servers.

**stat [-p period] [-r period] [-s separator] [-name filter] [-a attribute\_list] [-rate attribute\_list] [-delta attribute\_list] [-slope attribute\_pair\_list] [OUTPUT] [TARGET]**

- -p period : Sets the period in seconds. Default value is 10
- -r period : Sets the the period for rebuilding the list of MBeans to poll. Default value is 300
- -s separator : Sets the separator to delimit the obtained monitoring data. Default is ";"
- -name filter : Defines the MBeans to poll. FILTER may be an exact match of a MBean or contains wildcards according to the JMX conventions.

Example : stat -name "joramClient:type=queue,\*"

- **-a attribute\_list** : Sets the list of attributes to poll. If no attribute is specified, the attribute list is guessed from the first queried MBean.
- **-rate attribute\_list** : Option allowing to calculate the rate from the given attributes (dX/dT).
- **-delta attribute\_list** : Option allowing to calculate the delta from the given attributes (dX).
- **-slope attribute\_pair\_list** : Option allowing to calculate the slope from the given attributes (dX/dY). A pair number of attributes must be specified here, since they will be interpreted 2 by 2.
- **TARGET**

See Section 3.2, “Defining targets”

- **OUTPUT**

See Section 3.3, “Defining outputs”

## 3.6. The JDBC Connections command

**jdbccconnections [-o file] [-d duration] [-n datasource-name] [TARGET]**

- **-o -output-file <filename>**: Print the result in the given file.
- **-d -duration-filter <duration>**: Filter connection opened for more than <duration-filter> seconds.
- **-n -datasource-name <name>**: ObjectName of the DataSource to be observed.
- **TARGET**

See Section 3.2, “Defining targets”

Unless specified by the jasmine.jmx.url property, the default target is :

- `service:jmx:rmi:///jndi/rmi://localhost:1099/jrmpconnector_jonas`

### Example 3.4. JDBC Connection status

- from resource service

```
java -jar mbeancmd.jar jdbccconnections -n
jonas:j2eeType=JCAConnectionFactory,name=jdbc_1,JCAResource=db_access_jdbc1,J2EEServer=jonas
```

- from dbm service

```
java -jar mbeancmd.jar jdbccconnections -n
jonas:j2eeType=JDBCDataSource,name=HSQL1,JDBCResource=JDBCResource,J2EEServer=jonas
```



### Note

If the target server has secured jmx, the MBeanCmd client must have read-write permission.

## 3.7. The Dump command

**dump [-f file] [-l] [TARGET]**

- -f, -file filename : Prints thread stack dump in a given file "filename"
- -l, -log : Prints thread stack dump in JOnAS log file (level: info)

Example : dump -f /home/username/threaddump.txt

- TARGET

See Section 3.2, "Defining targets"

## 3.8. The Snap command

snap -name FILTER -a attributes [ -s attributes ] [ -delta [ time\_interval ] ] [ -z ] [ -m up | down ] [TARGET]

This command is useful to figure out what is going on on a set of JOnAS (or JVM instances). Use this command if you have no preconceived idea of what instance does what. It enables for example to pinpoint which servlets of which JOnAS instances are intensively requested, which EJBs take the most processing time, which instances have an intense transactionnel activity, which JVM have a high workload to do (many active threads). it can be used to audit a production site before setting up further stat or poll commands.

- -name FILTER : a mbean pattern, as defined by the JMX Specification. Example: to collect data on servlets, specify the FILTER as "\*:j2eeType=Servlet,\*".
- -a attributes : the names of the attributes to collect.
- -s attributes : the names of the attributes used for sorting the output. If absent, the first attribute of the -a option is used for sorting. If several attributes are supplied, they will be successively (and independently) used, and the command will print as many reports as "sort" attributes specified by the -s option.

### Example 3.5. The Snap -s option

On a "freshly started" JOnAS instance, where an administrator has connected to through the jonasAdmin console, the following command :

```
snap -name "*:j2eeType=Servlet,*" -a requestCount processingTime
```

produces the following output :

```
requestCount;source;mbean
27;j01;jasmine01:j2eeType=Servlet,name=default,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
3;j01;jasmine01:j2eeType=Servlet,name=action,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
1;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.welcomeContent_jsp,WebModule=//
localhost/jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
1;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.index_jsp,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
1;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.frameright_jsp,WebModule=//
localhost/jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
```

whereas the following command :

```
snap -name "*:j2eeType=Servlet,*" -a requestCount processingTime -s requestCount
processingTime
```

produces the following output (it first prints a list of mbeans by decreasing requestCount, then by decreasing processingTime):

```
requestCount;source;mbean
27;j01;jasmine01:j2eeType=Servlet,name=default,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
3;j01;jasmine01:j2eeType=Servlet,name=action,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
1;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.welcomeContent_jsp,WebModule=//
localhost/jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
1;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.index_jsp,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
1;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.frameright_jsp,WebModule=//
localhost/jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01

processingTime;source;mbean
2494;j01;jasmine01:j2eeType=Servlet,name=action,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
370;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.index_jsp,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
100;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.welcomeContent_jsp,WebModule=//
localhost/jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
70;j01;jasmine01:j2eeType=Servlet,name=default,WebModule=//localhost/
jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
60;j01;jasmine01:j2eeType=Servlet,name=org.apache.jsp.frameright_jsp,WebModule=//
localhost/jonasAdmin,J2EEApplication=none,J2EEServer=jasmine01
```

- **-delta [ time\_interval ]** : this option affects the snap behaviour. The command takes a baseline snapshot, then wait for time\_interval seconds (default is 60 seconds), then takes a current snapshot. Then it computes the differences between the current and baseline snapshots, and orders according to the differences rather than the current values.

### Example 3.6. The Snap -d option

Assume that within a slice of one minute, only the earsample application has been used. The following command :

```
snap -name "*:j2eeType=Servlet,*" -a requestCount processingTime -s requestCount
processingTime -delta
```

produces the following output :

```
requestCount;source;mbean
84.0;j01;jasmine01:j2eeType=Servlet,name=default,WebModule=//
localhost/earsample,J2EEApplication=earsample,J2EEServer=jasmine01
19.0;j01;jasmine01:j2eeType=Servlet,name=Op,WebModule=//localhost/
earsample,J2EEApplication=earsample,J2EEServer=jasmine01

processingTime;source;mbean
1012.0;j01;jasmine01:j2eeType=Servlet,name=Op,WebModule=//localhost/
earsample,J2EEApplication=earsample,J2EEServer=jasmine01
30.0;j01;jasmine01:j2eeType=Servlet,name=default,WebModule=//
localhost/earsample,J2EEApplication=earsample,J2EEServer=jasmine01
```

- -z : the snap command does not print zero value, unless this option is set.
- -m up | down : set the ordering mode: by increasing values (up), or by decreasing values (down). Default mode is down.
- TARGET

See Section 3.2, "Defining targets"

## 3.9. The Scan command

scan -a attributes [ -i path ] [ -current | -delta | -stat ] [ -z ] [ -m up | down ] [ -s separator ]

This command serves the same purpose as the snap command, but unlike snap, it is for offline usage. It processes the output of the poll or stat command, either from the standard input or from a file. The first line of the input (or file) is interpreted as a header of semicolon (";") separated field (or attribute) names. Subsequent lines are parsed and field values are retrieved from their position.

A metric corresponds to a record in the input (that is, a line). It is identified by the source (that represents a JVM instance), a mbean name within the source, and comprises of a set of attributes (or fields). The source is the "sname" field, the mbean is the "mbean" field. Timestamp are retrieved from the "time" field.

- -a attributes : the names of the attributes (or field) to collect.
- -i path : forces scan to process a file rather than the standard input.
- -s separator : forces scan to use the supplied field separator instead of the default one (semicolon ";").
- -z : the scan command does not print zero value, unless this option is set.
- -m up | down : set the ordering mode: by increasing values (up), or by decreasing values (down). Default mode is down.
- -current : The command processes the input and retains the most recent values for each metric. This option is useful when attributes are counters.
- -delta : the command builds a baseline from the first values retrieved from the input, then scan the input and retains the most recent values, then computes the variation from the baseline. this option is useful when attributes are counters.



- **-stat** : Processes the input and computes statistics for each attribute. Sort on mean values. Computed statistics are:
  - the average value for the whole scan.
  - the average value of non null values.
  - the minimal non null value.
  - the maximal value.
  - the time interval of the scan.
  - the time where values are not null during the scan.
  - the ratio between both times (  $\leq 1$  ).
 This option is useful for both counter and gauges attributes.



### Note

The **-current**, **-delta** and **-stat** options are mutually exclusive.

### Example 3.7. The Scan command

The following command

```
scan -i tx.log -a commit
```

produces the following output:

```
commit;range;source;mbean
56996.0;4.8;jonas01;jonas01:J2EEServer=jonas01,j2eeType=JTAResource,name=JTAResource
24546.0;4.4;jonas02;jonas02:J2EEServer=jonas02,j2eeType=JTAResource,name=JTAResource
5758.0;3.8;jonas03;jonas03:J2EEServer=jonas03,j2eeType=JTAResource,name=JTAResource
```

### Example 3.8. The Scan command : -delta option

The following command (with the **-delta** option)

```
scan -i tx.log -a commit -delta
```

produces the following output:

```
commit;range;source;mbean
47199.0;4.7;jonas01;jonas01:J2EEServer=jonas01,j2eeType=JTAResource,name=JTAResource
4095.0;3.6;jonas02;jonas02:J2EEServer=jonas02,j2eeType=JTAResource,name=JTAResource
```

jonas03 is no more listed, as no transaction has been committed on it while the tx.log file was recorded.

## 3.10. The mbean command

This command offers a low level interface to access MBeans. It is dedicated to expert administrators.

**mbean [-v] -name FILTER -ACCESS [ACCESS-OPTS] [TARGET]**

- `-v` : Produces a verbose output.
- `-name FILTER` : Defines the MBeans to access. The `FILTER` may be an exact match of a MBean name (`OBJECT_NAME`), or may contains wildcards according to the JMX conventions.

Example : `mbean -name "jonas:j2eeType=JVM,*" -query`

- `-ACCESS [ACCESS-OPTS]` : Defines the type of access. Examples of access types are: `-query` and `-info`. Note that the `-info` access needs an exact matching `OBJECT_NAME`.

You may get details on all the access types and corresponding options by using the help command

```
help mbean
```

- `TARGET`

See Section 3.2, “Defining targets”

### Example 3.9. MBean command examples

Suppose that administrator needs to access the **JVM** MBean in a JOnAS server named **jonas5**.

```
mbean -name "*:j2eeType=JVM,*" -query
```

```
mbean -name "jonas:j2eeType=JVM,name=jonas5,J2EEServer=jonas5" -info
```

```
mbean -name "jonas:j2eeType=JVM,name=jonas5,J2EEServer=jonas5" -get allThreadsCount
```

## 3.11. The WildStat command

**wildstat [-p period] [-r period] [-s separator] BEAN [-a attributes] [-c config] [-q query] [OUTPUT] [TARGET]**

This command allows you to launch an MBeanCmd command using the features offers by WildCAT [<http://wildcat.ow2.org>]. WildCAT permits to build a hierarchical context in which you can organize datas and query for operations like aggregation and filtering.



### Note

This command is only available in the distribution `mbeancmd-wildstat`. This is due to a licence limitation of Esper [<http://esper.codehaus.org/>], the Complex Event Processing engine on which WildCAT is based.

- `-p period` : Sets the period in seconds.
- `-r period` : Sets the the period for rebuilding the list of mbeans to poll. Default refresh period is 300 seconds.
- `-s separator` : Sets the separator. Default is `","`.
- **BEAN**
  - `-name filter` : Defines the mbeans to poll. `filter` may be an exact match of a mbean or contains wildcards according to the JMX conventions.
- `-a attributes` : Sets the list of attributes to poll. If no attribute is specified, the attribute list is guessed from the first queried mbean.

- `-c config` : Sets the configuration file for WildCAT.
- `-q query` : Sets an Event Query to aggregate or filter results. Notice that aggregated fields have to be the name prefixed by "my". default : `select * from org.ow2.jasmine.event.attribute.JasmineAttributeChangedEvent`



## Note

the query language is EPL (Esper Processing Language). You can find documentation on it here [[http://esper.codehaus.org/esper-2.3.0/doc/reference/en/html\\_single/index.html#epl\\_clauses](http://esper.codehaus.org/esper-2.3.0/doc/reference/en/html_single/index.html#epl_clauses)].

## OUTPUT

The command prints statistics into stdout, unless the `-f` or `-jasmine` option is set. The `-f`, `-graph` and `-jasmine` options are not exclusive.

- `-export` : export the result in the WildCAT context defined in the configuration file (cf. `-c` option). The result will be sent in `self://jasmine/mbeancmd#wildstat`.
- `-f FILE` : Causes the output to be redirected to FILE.
- `-graph XML_DEF` : Displays graphs according to the XML definitions specified by the XML\_DEF file.
- `-jasmine URL` : Causes the output to be redirected to the JASMINe Event Switch located at URL.

## TARGET

See Section 3.2, "Defining targets"

## Example 3.10. The WildStat command examples

`wildstat -p 10 -r 300 -name "joramClient:type=queue,*" -c standalone.properties -q "select * from org.ow2.jasmine.event.attribute.JasmineAttributeChangedEvent(value>=100)" -target all -jasmine vm://JasmineEventDispatcher`

`wildstat -q "select domain, server, objectName, probe, sname, avg(cast(value,int)) as value from org.ow2.jasmine.event.attribute.JasmineAttributeChangedEvent.win:time(10 sec) group by domain, server, objectName, probe, sname"`

## 3.12. Example: output the number of threads

If we browse the JOnAS Administration interface, we see that the JVM J2EE Bean (called in our case **jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas**) has an attribute called **allThreadsCount** (The number of active threads in the JVM).

The screenshot shows the JOnAS Administration console. On the left is a tree view of the system components. The main panel displays the details for the bean `jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas`. Under the 'Attributes' tab, a table lists various attributes. The attribute `allThreadsCount` is highlighted with a red box, indicating its value is 110. Other attributes include `javaVendor`, `javaVersion`, `modelType`, `node`, `objectName`, `stateManageable`, `statisticsProvider`, and `threadsGroups`.

Attribute	Value
<code>allThreadsCount</code>	110
<code>javaVendor</code>	Java HotSpot(TM) Client VM-1.5.0_14-b03 / Sun Microsystems Inc.
<code>javaVersion</code>	1.5.0_14
<code>modelType</code>	org.objectweb.jonas.server.JVM
<code>node</code>	FRECB00380.adch.fr:bull
<code>objectName</code>	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas
<code>stateManageable</code>	false
<code>statisticsProvider</code>	false
<code>threadsGroups</code>	[System-main:RMI Runtime-HSQLDB Connectors @1808914]

We'll now output this information on different mediums.

### 3.12.1. On the console

We know that:

- We'll be targeting only one server
- The IP address of the targeted server is 129.183.101.109
- The JOnAS Administration interface tells the JRMP MBean connector is on service:jmx:rmi:///jndi/rmi://localhost:1099/jrmpconnector\_jonas
- We want a refresh period of 1 second

Therefore, the command to launch is:

```
java -Djasmine.jmx.url=service:jmx:rmi:///jndi/rmi://129.183.101.109:1099/
jrmpconnector_jonas -jar mbean.jar stat -p 1 -name "*:j2eeType=JVM,*" -a allThreadsCount
```

This displays the following output:

```
date;time;sname;server;domain;mbean;allThreadsCount
2007/12/21
11:03:22;1198231402118;jonas;jonas;jonas;jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas;125
2007/12/21
11:03:23;1198231403194;jonas;jonas;jonas;jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas;126
2007/12/21
11:03:24;1198231404254;jonas;jonas;jonas;jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas;123
2007/12/21
11:03:25;1198231405313;jonas;jonas;jonas;jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas;130
2007/12/21
11:03:26;1198231406373;jonas;jonas;jonas;jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas;126
[etc]
```

On that output, you'll notice that:

- The first row of data contains the list of attributes
- All other rows contain the sampled data using the format specified in the first row

### 3.12.2. On a spreadsheet

Since any spreadsheet software can read comma separated values, it is rather trivial to import data captured by MBeanCmd into a spreadsheet: just copy and paste the console output:

	A	B	C	D	E	F	G
1	date	time	sname	server	domain	mbean	allThreadsCount
2	21/12/2007 10:45	1.19823E+12	1	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	110
3	21/12/2007 10:45	1.19823E+12	2	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	111
4	21/12/2007 10:45	1.19823E+12	3	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	112
5	21/12/2007 10:45	1.19823E+12	1	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	113
6	21/12/2007 10:45	1.19823E+12	2	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	114
7	21/12/2007 10:45	1.19823E+12	3	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	115
8	21/12/2007 10:45	1.19823E+12	1	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	116
9	21/12/2007 10:45	1.19823E+12	2	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	117
10	21/12/2007 10:45	1.19823E+12	3	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	118
11	21/12/2007 10:45	1.19823E+12	1	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	119
12	21/12/2007 10:45	1.19823E+12	2	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	120
13	21/12/2007 10:45	1.19823E+12	3	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	121
14	21/12/2007 10:45	1.19823E+12	1	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	122
15	21/12/2007 10:45	1.19823E+12	2	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	123
16	21/12/2007 10:45	1.19823E+12	3	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	124
17	21/12/2007 10:45	1.19823E+12	1	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	125
18	21/12/2007 10:45	1.19823E+12	2	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	126
19	21/12/2007 10:45	1.19823E+12	3	jonas	jonas	jonas:j2eeType=JVM,name=jonas,J2EEServer=jonas	127

### 3.12.3. In a graph

The **-graph** option can be used to display the sampled data in a graph. The XML DTD of a graph can be accessed by typing:

```
java -jar mbean.jar help stat
```

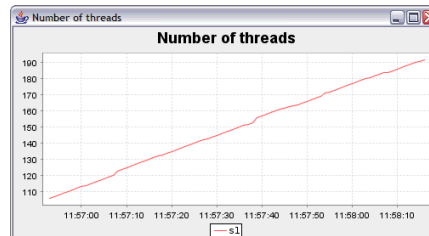
or

```
java -jar mbean.jar help poll
```

To display the number of threads in a simple graph, the following **graph.xml** file can be used:

```
java -Djasmine.jmx.url=service:jmx:rmi:///jndi/rmi://129.183.101.109:1099/  
jrmppconnector_jonas -jar mbean.jar stat -p 1 -graph graph.xml -name "*:j2eeType=JVM,*" -a  
allThreadsCount
```

Which will display the following graph:



Note that you can also display multiple series on one graph.

# Chapter 4. MGen - a configuration generator for MBeanCmd

## 4.1. Pre-Requisites

We will assume that we want to monitor a JOnAS cluster, and that we have got a working compiled version of MBeanCmd in the following path : \$MGEN\_ROOT/../../dist/mbean.jar.

## 4.2. MGen setup

### 4.2.1. Tree structure

```
-- build-components.xml
-- build.xml
-- mgen-conf.xml
-- output
| -- dist
| | -- mbean.jar
| -- graphs
| | -- cpusun.xml
| | -- jmxurls.properties
| | -- monitoring.cmd
| | -- monitoring.sh
| | -- reports
| -- scripts
| | -- cpusun.cmd
| | -- cpusun.sh
| | -- mbeancmd.cmd
| | -- mbeancmd.sh
-- styles
| -- mgen-ant.xml
| -- mgen-graph.xml
| -- mgen-jmxurl.xml
| -- mgen-script-unix-monitoring.xml
| -- mgen-script-unix.xml
| -- mgen-script-win-monitoring.xml
| -- mgen-script-win.xml
-- templates
| -- mbeancmd.cmd
| -- mbeancmd.sh
```

Here stands the tree structure of the "mgen" repertory.

- The mgen-conf.xml file enables you to set the details you want to monitor. In the following example, the CPU load is switched on.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<mgen-conf>
  <global>
    <vars>
      <var id="APPCTX">/sampleCluster2</var>
      <var id="J2EEAPP">sampleCluster2</var>
      <var id="DSAPP">jdbc_1</var>
    </vars>

    <components>
      <component id="cpusun">
        <cmd-unix>poll -cpusun</cmd-unix>
        <cmd-win>poll -cpusun</cmd-win>
        <series>
          <serie id="currentCpuLoad" title="currentCpuLoad" col="currentCpuLoad"
type="double"/>
        </series>
        <graphs>
          <graph id="g1" title="CPU Load">
            <serie id="currentCpuLoad"/>
          </graph>
        </graphs>
      </component>
    </components>
  </global>
</mgen-conf>
```

```

    </component>
  </components>
</global>

<servers>
  <server>
    <name>jonas</name>
    <host>localhost</host>
    <port>1099</port>
    <protocol>jrmp</protocol>
  </server>
</servers>
</mgen-conf>

```

The "components" section describes all the indicators we want to supervise. For each, the unix and windows MBeanCmd commands are described, as well as the description of the graph. The main part of the section is the "series" one, where we can define which indicators we want to monitor. Finally, the "servers" section is where we setup on which JOnAS server we are going to do the monitoring.

- The "output" directory contains all the files generated by the ant script. In the "dist" directory stands the file mbean.jar, which is a simple copy of the file \$MGEN\_ROOT/./dist/mbean.jar. The "graphs" directory contains the xml graph description files parsed by the grapher to make all the graphs we asked for. "monitoring.sh" script launch all the mbeancmd scripts contained in the "scripts" repertory. We can obviously launch those scripts one by one.
- The "styles" repertory contains all the xsl files used for generating the configuration.
- The "templates" directory, where mbeancmd environment variables are contained.

## 4.2.2. Compile MGen

Nothing to be scared of, just run "ant" in a command line. This will ask us for the repertory name to put the result in (default is ./output/). Generated files have been seen in the previous section.

## 4.3. mgen-conf.xml ... scheme summary of THE file

### 4.3.1. "vars" section

Here we can define some variables we may require in the next sections. Typically, the variable may be associated with the application context. Just take a look on the following example :

```

<?xml version='1.0' encoding='ISO-8859-1' ?>
<mgen-conf>
  <global>
    <vars>
      <var id="APPCTX">./sampleCluster2</var>
    </vars>

    [...]

  <components>
    <component id="session">
      <cmd-unix>stat -name *:type=Manager,path=$APPCTX,* -a name activeSessions
sessionCounter expiredSessions</cmd-unix>
      <cmd-win>stat -name &quot;*:type=Manager,path=%APPCTX%,*&quot; -a name
activeSessions sessionCounter expiredSessions</cmd-win>
    </component>
    [...]
  </components>
</mgen-conf>

```

The variable APPCTX will be replaced by its value when the file will be parsed. We can define as many variables as we want to.

### 4.3.2. "components" section

This is where all the components we want to monitor are defined. Here is a good example of what a component description could be :

```
<component id="tx">
  <cmd-unix>poll -tx</cmd-unix>
  <cmd-win>poll -tx</cmd-win>

  <series>
    <serie id="txRate" title="Rate" col="txRate" type="double"/>
    <serie id="pending" title="Pending" col="pending" type="double"/>
    <serie id="commit" title="Commits" col="commit" type="double"/>
    <serie id="rollback" title="Rollbacks" col="rollback" type="double"/>
    <serie id="timedOut" title="Timeouts" col="timedOut" type="double"/>
    <serie id="commitRatio" title="Mean commit ratio" col="commitRatio" type="double"/>
    <serie id="currentCommitRatio" title="Current commit ratio" col="currentCommitRatio"
type="double"/>
  </series>

  <graphs>
    <graph id="g1" title="Transactions en cours">
      <serie id="txRate"/>
      <serie id="pending"/>
    </graph>
    <graph id="g2" title="Rollbacks">
      <serie id="rollback"/>
      <serie id="timedOut"/>
    </graph>
    <graph id="g3" title="Commit ratio">
      <serie id="commitRatio"/>
      <serie id="currentCommitRatio"/>
    </graph>
    <graph id="g4" title="Nb Transactions">
      <serie id="commit"/>
    </graph>
  </graphs>
</component>
```

Each component is identified with a unique id. There are then three main parts :

- The first part is where the commands of MBeanCmd are set, both for unix and windows systems.
- The second part deals with the series to get from the option given to MBeanCmd. A serie should have an unique "id", a "title", a "col" name and finally the type that will be set to expose the data.
- Finally, the "graphs" section defines the graph we want to draw, according to the series set before. In this example, four graphs will be drawn.

### 4.3.3. "servers" section

This is the last section of the file, where the servers to connect on are set. The following example shows two JOnAS server connections, one with authentication and the other without. We can define the "name" of the node, the "host" and the "port" number as well as the "protocol" to use. "user" and "password" are to be used only if the JMX authentication is set.

```
<servers>
  <server>
    <name>node1</name>
    <host>localhost</host>
    <port>2002</port>
    <protocol>jrmp</protocol>
  </server>
  <server>
```



```
<name>node2</name>
<host>localhost</host>
<port>2022</port>
<protocol>jrmp</protocol>
<user>jonas</user>
<password>jonas</password>
</server>
</servers>
```