# 14 JMX monitoring

## 14.1 Overview

JMX monitoring can be used to monitor JMX counters of a Java application.

In Zabbix 1.8, if you wanted to monitor JMX counters of a Java application, your best choice would have been the Zapcat JMX Zabbix Bridge. You would either modify the source code of your application to reference the Zapcat JAR file and programmatically start a Zabbix agent, or you would install a ready-made Zapcat plugin for applications that support it (such as Jetty or Tomcat).

Zabbix 2.0 adds native support for JMX monitoring by introducing a new Zabbix daemon called "Zabbix Java gateway".

When Zabbix server wants to know the value of a particular JMX counter on a host, it asks the Zabbix **Java gateway**, which in turn uses the JMX management API to query the application of interest remotely.

For more details on Zabbix Java gateway, including where to get it and how to set it up see this section of the manual.

## 14.2 Enabling remote JMX monitoring for Java application

A Java application does not need any additional software installed, but it needs to be started with the command-line options specified below to have support for remote JMX monitoring.

As a bare minimum, if you just wish to get started by monitoring a simple Java application on a local host with no security enforced, start it with these options:

```
java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

This makes Java listen for incoming JMX connections on port 12345 and tells it not to require authentication or SSL.

If you wish to be more stringent about security, there are many other Java options available to you. For instance, the next example starts the application with a more versatile set of options and opens it to a wider network, not just local host.

```
java \
-Djava.rmi.server.hostname=192.168.3.14 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/etc/java-6-
openjdk/management/jmxremote.password \
```

```
-Dcom.sun.management.jmxremote.access.file=/etc/java-6-
openjdk/management/jmxremote.access \
-Dcom.sun.management.jmxremote.ssl=true \
-Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

Most (if not all) of these settings can be specified in /etc/java-6-openjdk/management/management.properties (or wherever that file is on your system).

Note that if you wish to use SSL, you have to modify startup.sh script by adding `-Djavax.net.ssl.*` options to Java gateway, so that it knows where to find key and trust stores.

See Monitoring and Management Using JMX for a detailed description.

## 14.3 Configuring JMX interfaces and items in Zabbix GUI

With Java gateway running, server knowing where to find it and a Java application started with support for remote JMX monitoring, it is time to configure the interfaces and items in Zabbix GUI.

#### Configuring JMX interface

You begin by creating a JMX-type interface on the host of interest:

#### Adding JMX agent item

For each JMX counter you are interested in you add an item of type **JMX agent** attached to that interface. If you have configured authentication on your Java application, then you also specify username and password.

The key is not seen fully in the screenshot below, but it says `jmx["java.lang:type=Memory","HeapMemoryUsage.used"]`. The JMX item key syntax is similar to Zapcat items, except that a comma is used for separating arguments instead of "][". The key consists of

- object name - which represents the object name of an MBean
- attribute name - an MBean attribute name with optional composite data field names separated by dots

See below for examples of JMX item keys.

If you wish to monitor a Boolean counter that is either “true” or “false”, then you specify type of information as “Numeric (unsigned)” and data type as “Boolean”. Server will store Boolean values as 1 or 0, respectively.

#### Examples of JMX item keys

An attribute may return a primitive data type (string, integer etc.) in that case a key looks like this:

```
jmx[com.example:type=Hello,myattribute]
```

If an attribute name contains dots they need to be escaped with a backslash:

```
jmx[com.example:type=Hello,my\.attribute\.with\.dots]
```

Because in case an attribute returns a composite data a dot is used to separate an attribute name and composite data field names:

```
jmx[com.example:type=Hello,myattribute.car.weight] ("myattribute" -> "car"
-> "weight")
jmx[com.example:type=Hello,my\.attribute\.with\.dots.car.weight\.of\.the\.ca
r] ("my.attribute.with.dots" -> "car" -> "weight.of.the.car")
```

A backslash character should be escaped as well:

```
jmx[com.example:type=Hello,c:\\documents]
```

If the name contains special symbols like spaces, commas double-quote it:

```
jmx["java.lang:name=ConcurrentMarkSweep,type=GarbageCollector","LastGcInfo.m
emoryUsageAfterGc.CMS Old Gen.committed"]
```

This is actually all there is to it. Happy JMX monitoring!