4 Java gateway

Overview

Native support for monitoring JMX applications exists in the form of a Zabbix daemon called "Zabbix Java gateway", available since Zabbix 2.0. Zabbix Java gateway is a daemon written in Java. To find out the value of a particular JMX counter on a host, Zabbix server queries Zabbix Java gateway, which uses the JMX management API to query the application of interest remotely. The application does not need any additional software installed, it just has to be started with `-Dcom.sun.management.jmxremote` option on the command line.

Java gateway accepts incoming connection from Zabbix server or proxy and can only be used as a "passive proxy". As opposed to Zabbix proxy, it may also be used from Zabbix proxy (Zabbix proxies cannot be chained). Access to each Java gateway is configured directly in Zabbix server or proxy configuration file, thus only one Java gateway may be configured per Zabbix server or Zabbix proxy. If a host will have items of type **JMX agent** and items of other type, only the **JMX agent** items will be passed to Java gateway for retrieval.

When an item has to be updated over Java gateway, Zabbix server or proxy will connect to the Java gateway and request the value, which Java gateway in turn retrieves and passes back to the server or proxy. As such, Java gateway does not cache any values.

Zabbix server or proxy has a specific type of processes that connect to Java gateway, controlled by the option **StartJavaPollers**. Internally, Java gateway starts multiple threads, controlled by the **START_POLLERS** option. On the server side, if a connection takes more than **Timeout** seconds, it will be terminated, but Java gateway might still be busy retrieving value from the JMX counter. To solve this, since Zabbix 2.0.15, Zabbix 2.2.10 and Zabbix 2.4.5 there is the **TIMEOUT** option in Java gateway that allows to set timeout for JMX network operations.

Zabbix server or proxy will try to pool requests to a single JMX target together as much as possible (affected by item intervals) and send them to the Java gateway in a single connection for better performance.

It is suggested to have **StartJavaPollers** less than or equal to **START_POLLERS**, otherwise there might be situations when no threads are available in the Java gateway to service incoming requests.

Sections below describe how to get and run Zabbix Java gateway, how to configure Zabbix server (or Zabbix proxy) to use Zabbix Java gateway for JMX monitoring, and how to configure Zabbix items in Zabbix GUI that correspond to particular JMX counters.

1 Getting Java gateway

You can install Java gateway from either the downloaded packages or sources on the Zabbix website.

See instructions for Java gateway installation:

- from packages for RHEL/CentOS
- from packages for Debian/Ubuntu
- by compiling from sources

2 Overview of files in Java gateway distribution

Regardless of how you obtained Java gateway, you should have ended up with a collection of shell scripts, JAR and configuration files under $PREFIX/sbin/zabbix_java. The role of these files is summarized below.

```
bin/zabbix-java-gateway-$VERSION.jar
```

Java gateway JAR file itself.

```
lib/logback-core-0.9.27.jar
lib/logback-classic-0.9.27.jar
lib/slf4j-api-1.6.1.jar
lib/android-json-4.3_r3.1.jar
```

Dependencies of Java gateway: Logback, SLF4J, and Android JSON library.

```
lib/logback.xml
lib/logback-console.xml
```

Configuration files for Logback.

```
shutdown.sh
startup.sh
```

Convenience scripts for starting and stopping Java gateway.

```
settings.sh
```

Configuration file that is sourced by startup and shutdown scripts above.

3 Configuring and running Java gateway

By default, Java gateway listens on port 10052. If you plan on running Java gateway on a different port, you can specify that in settings.sh script. See the description of Java gateway configuration file for how to specify this and other options.

Port 10052 is not IANA registered.

Once you are comfortable with the settings, you can start Java gateway by running the startup script:

```
$ ./startup.sh
```

Likewise, once you no longer need Java gateway, run the shutdown script to stop it:

```
$ ./shutdown.sh
```

Note that unlike server or proxy, Java gateway is lightweight and does not need a database.

4 Configuring server for use with Java gateway

With Java gateway up and running, you have to tell Zabbix server where to find Zabbix Java gateway. This is done by specifying JavaGateway and JavaGatewayPort parameters in the server configuration file. If the host on which JMX application is running is monitored by Zabbix proxy, then you specify the connection parameters in the proxy configuration file instead.

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

By default, server does not start any processes related to JMX monitoring. If you wish to use it, however, you have to specify the number of pre-forked instances of Java pollers. You do this in the same way you specify regular pollers and trappers.

```
StartJavaPollers=5
```

Do not forget to restart server or proxy, once you are done with configuring them.

5 Debugging Java gateway

In case there are any problems with Java gateway or an error message that you see about an item in the frontend is not descriptive enough, you might wish to take a look at Java gateway log file.

By default, Java gateway logs its activities into /tmp/zabbix_java.log file with log level “info”. Sometimes that information is not enough and there is a need for information at log level “debug”. In order to increase logging level, modify file lib/logback.xml and change the level attribute of <root> tag to “debug”:

```
<root level="debug">
  <appender-ref ref="FILE" />
</root>
```

Note that unlike Zabbix server or Zabbix proxy, there is no need to restart Zabbix Java gateway after changing logback.xml file - changes in logback.xml will be picked up automatically. When you are done with debugging, you can return the logging level to “info”.

If you wish to log to a different file or a completely different medium like database, adjust logback.xml file to meet your needs. See Logback Manual for more details.

Sometimes for debugging purposes it is useful to start Java gateway as a console application rather than a daemon. To do that, comment out PID_FILE variable in settings.sh. If PID_FILE is omitted, startup.sh script starts Java gateway as a console application and makes Logback use lib/logback-console.xml file instead, which not only logs to console, but has logging level “debug” enabled as well.

Finally, note that since Java gateway uses SLF4J for logging, you can replace Logback with the framework of your choice by placing an appropriate JAR file in lib directory. See SLF4J Manual for more details.