Using Jolokia with Mule ESB

Mitch Dresdner

Table of Contents

***A fast easy guide to integrating Mule ESB with the Jolokia monitoring** agent.*

Monitoring made simple with Jolokia Mule plugin

Summary

Monitoring is an essential component for ensuring the proper functioning of Microservices. As the complexity of software systems increases, it becomes more difficult to understand performance characteristics and to troubleshoot problems. The monitoring tools we use may also need to be adaptible to new deployment strategies and environments.

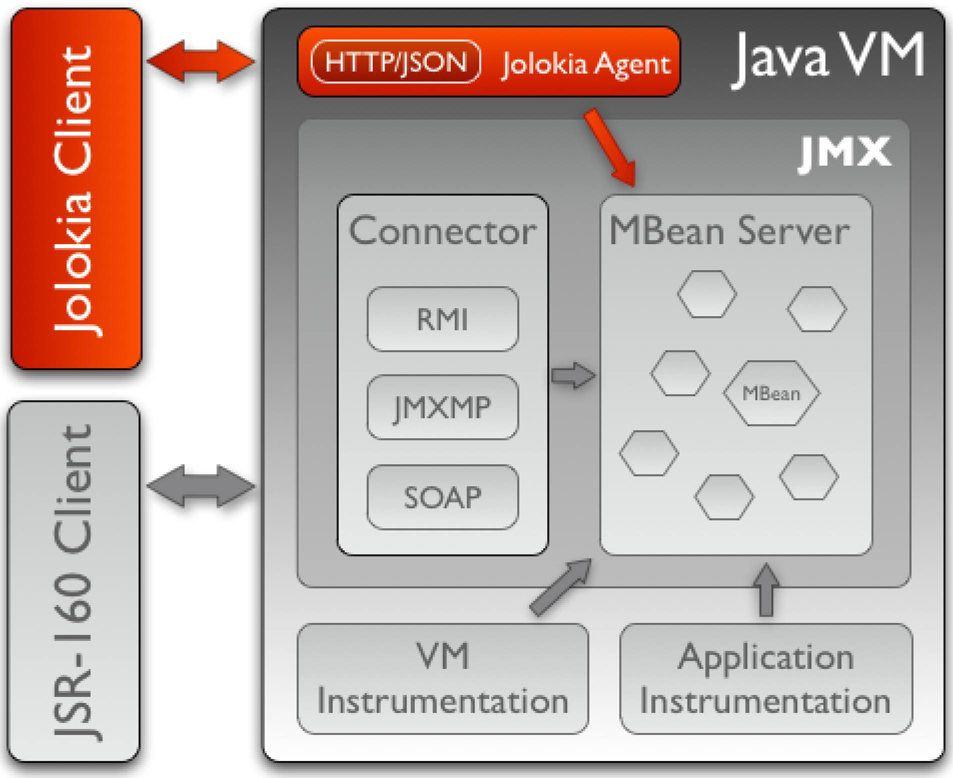
Jolokia is a JMX-HTTP bridge giving an alternative to JSR-160 connectors. It's an agent based approach with support for many platforms. In addition to basic JMX operations it enhances JMX remoting with unique features like bulk requests and fine grained security policies.

This paper describes how we might utilize Open Source Jolokia as a monitoring agent for Mule. The Mule ESB has historically used the Mule Management Console (MMC) to monitor internal flow and performance characteristics.

As we start pushing out the Mule runtime as a container based solution, we'll need to give careful thought to an appropriate monitoring strategy. Here we'll consider one, the Jolokia JMX Mule Agent plugin.

Architecture

The Jolokia client runs as a browser based instance from a Jar implemenmtation to communicate with the monitoring server we'll embed in the Mule runtime. The Jolokia Agent is able to communicate over various protocols such as RMI, SOAP and JMX depicted in the diagram below.



Mule monitoring information, propertied and statistics will be made available to the Jolokia Client through JMX Managed Beans (MBeans). The details are important if you plan to expose custom monitoring data to your clients.

For more information on how to customize your data see the Jokokia Documentation.

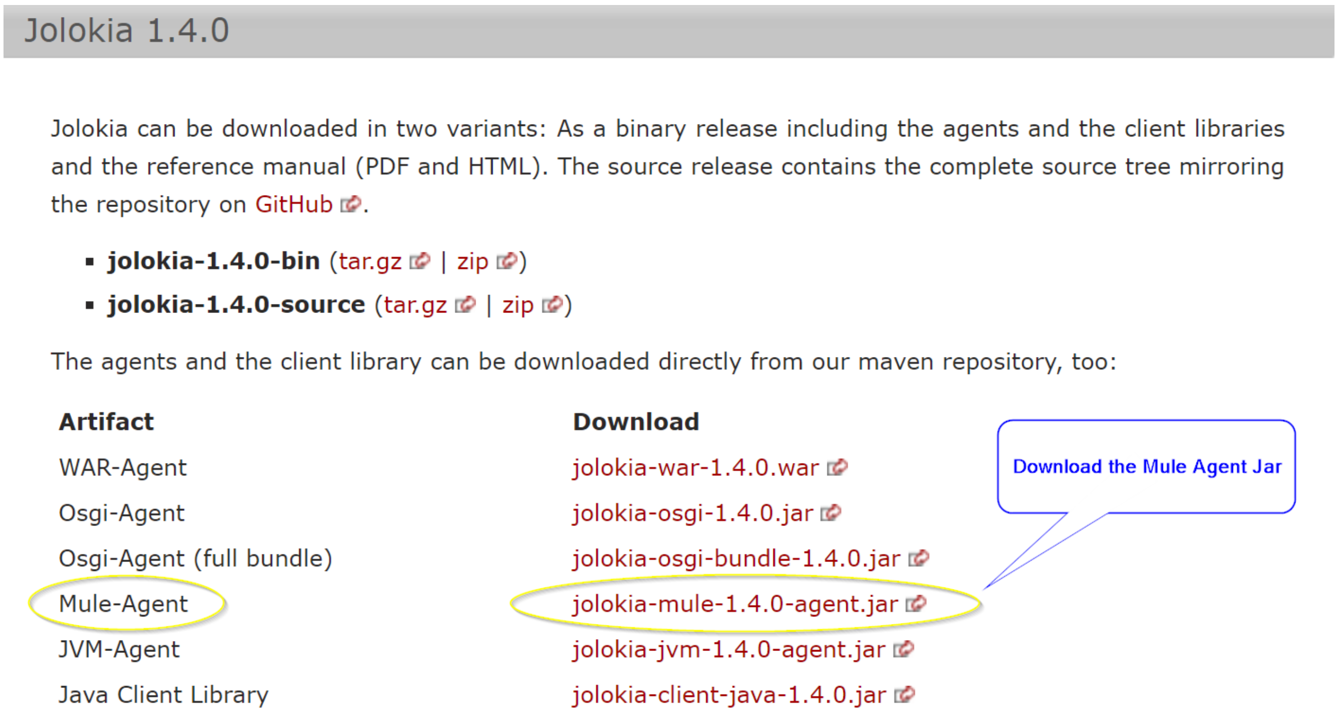
Installation

This installation will require two parts, the Jolokia Agent which we'll embed in the Mule Runtime and a client piece which will use Hawtio to provide browser access to the Jolokia agent.

Embedding Jolokia in Mule

Start by downloading the Agent - Jolokia Mule Download

The snapshot below shows the latest version I had pulled for this article, if the version has changed since then, download the latest version.


Jolokia 1.4.0

Jolokia can be downloaded in two variants: As a binary release including the agents and the client libraries and the reference manual (PDF and HTML). The source release contains the complete source tree mirroring the repository on GitHub.

- **jolokia-1.4.0-bin** (tar.gz | zip)
- **jolokia-1.4.0-source** (tar.gz | zip)

The agents and the client library can be downloaded directly from our maven repository, too:

Artifact	Download
WAR-Agent	jolokia-war-1.4.0.war
Osgi-Agent	jolokia-osgi-1.4.0.jar
Osgi-Agent (full bundle)	jolokia-osgi-bundle-1.4.0.jar
Mule-Agent	jolokia-mule-1.4.0-agent.jar
JVM-Agent	jolokia-jvm-1.4.0-agent.jar
Java Client Library	jolokia-client-java-1.4.0.jar

After downloading the Jolokia Mule Agent jar file, it needs to be stored in the Mule installation root folder, in $MULE_HOME/lib/opt/.

Next we'll embed the Jolokia Agent into the Mule default stub. Using your favorite editor go ahead and make the changes to the *mule-config.xml*, ensure that the XML namespace is consistent with the example.

Add the Jolokia Agent reference to the apps/default/mule-config.xml

```
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:management="http://www.mulesoft.org/schema/mule/management"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-current.xsd
             http://www.mulesoft.org/schema/mule/core
       http://www.mulesoft.org/schema/mule/core/current/mule.xsd
             http://www.mulesoft.org/schema/mule/management
       http://www.mulesoft.org/schema/mule/management/current/mule-management.xsd">

    ①
   <!-- Wire in Mule Jolokia Agent -->
   <custom-agent name="jolokia-agent" class="org.jolokia.mule.JolokiaMuleAgent">
      <spring:property name="port" value="8899"/>

    ②
      <!-- Uncomment to enable access control

        <spring:property name="user" value="mulokia"/>
        <spring:property name="password" value="secret"/>

        -->

   </custom-agent>
</mule>
```

① Add the reference for the Jolokia Agent in

② To enable access control

With the mule-config.xml file updated with the reference to the **jolokia-agent**, we can open the firewall port the agent listens on and send a request from the browser.

Agent Port

```
Open port 8899 for Jolokia agent
```

The procedure for opening the port will be different depending on whether your application is running in AWS, a Docker container, standalone Unix instance or your desktop/laptop. I’ll leave it as an exercise for you to solve with Google, if you really need help add a comment with your patform information and what you’ve tried.

Download Hawtio client

Use the link below to download the Hawtio Application from the link below.

Download the Hawtio App

Whew, that was easy! Let move on to configuring and starting.

Configure Hawtio

Lets start by configuring a script that will start up the Hawtio. Using whatever process you use to find your support Jars, add the Hawtio App jar you just downloaded to that location. Here's an example script I use to start the Hawtio App:

Hawtio batch file hawt.bat

```
%echo off%
set HAWT_PATH=/Tools/Java/Jars/hawtio-app-1.5.4.jar ①
set HAWT_PORT=8087②

java  -Dhawtio.proxyWhitelist=* -Dhawtio.authenticationEnabled=false -jar %HAWT_PATH%
--port %HAWT_PORT%
```

① I keep my non Maven/Gradle service Jars in c:\Tools\java\Jars, you can use Unix file path naming conventions and Java will find them

② Pick a free port number for Jetty to use for the Hawtio application

These same notes apply to the Bash script for Unix users below.

Hawtio script file hawt

```
#!/bin/bash
export HAWT_PATH=$HOME/Java/Jars/hawtio-app-1.5.4.jar
export HAWT_PORT=8087

java  -Dhawtio.proxyWhitelist=* -Dhawtio.authenticationEnabled=false -jar $HAWT_PATH
--port $HAWT_PORT
```

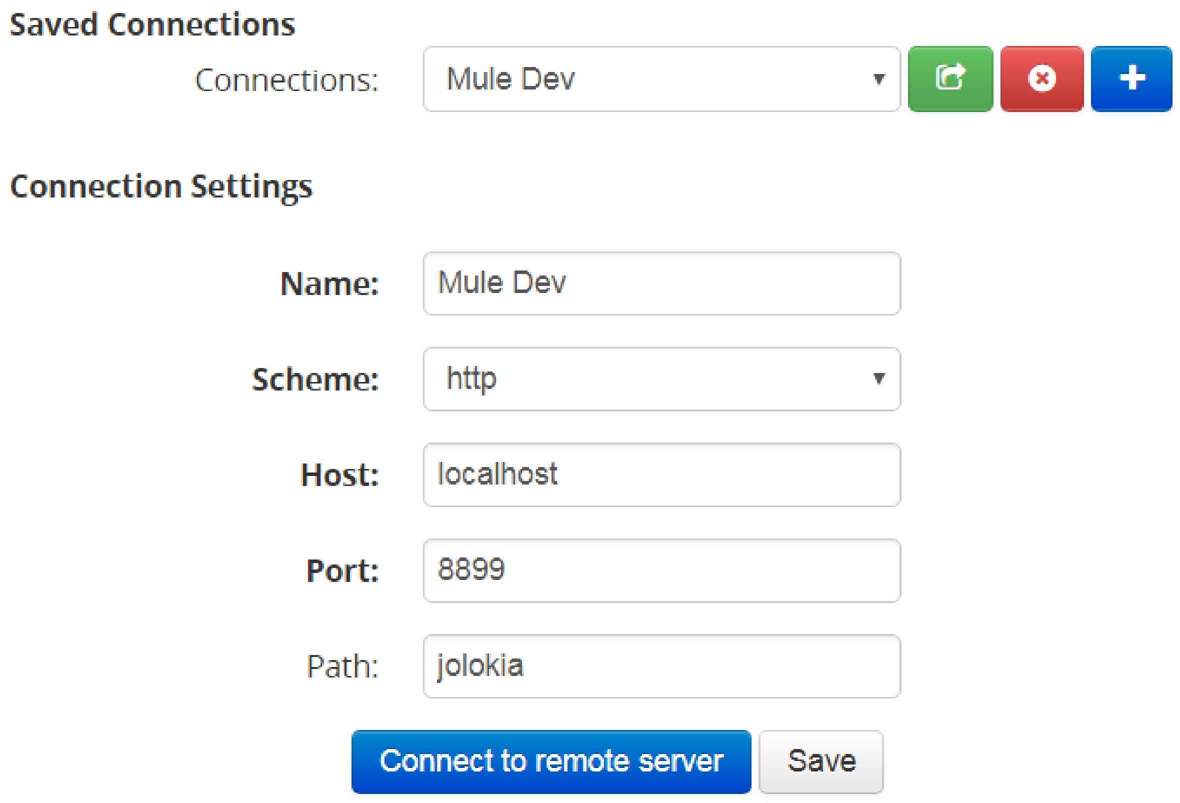
Starting Hawtio

Hawtio will start up a Jetty server on the port you specified in the configuration, then it will start a browser session connecting to the startup URI.

Run your script

```
hawt
```

The **Welcome** page serves to get beginners oriented with Hawtio, click on the **Connect** menu option.



Saved Connections

Connections:	Mule Dev

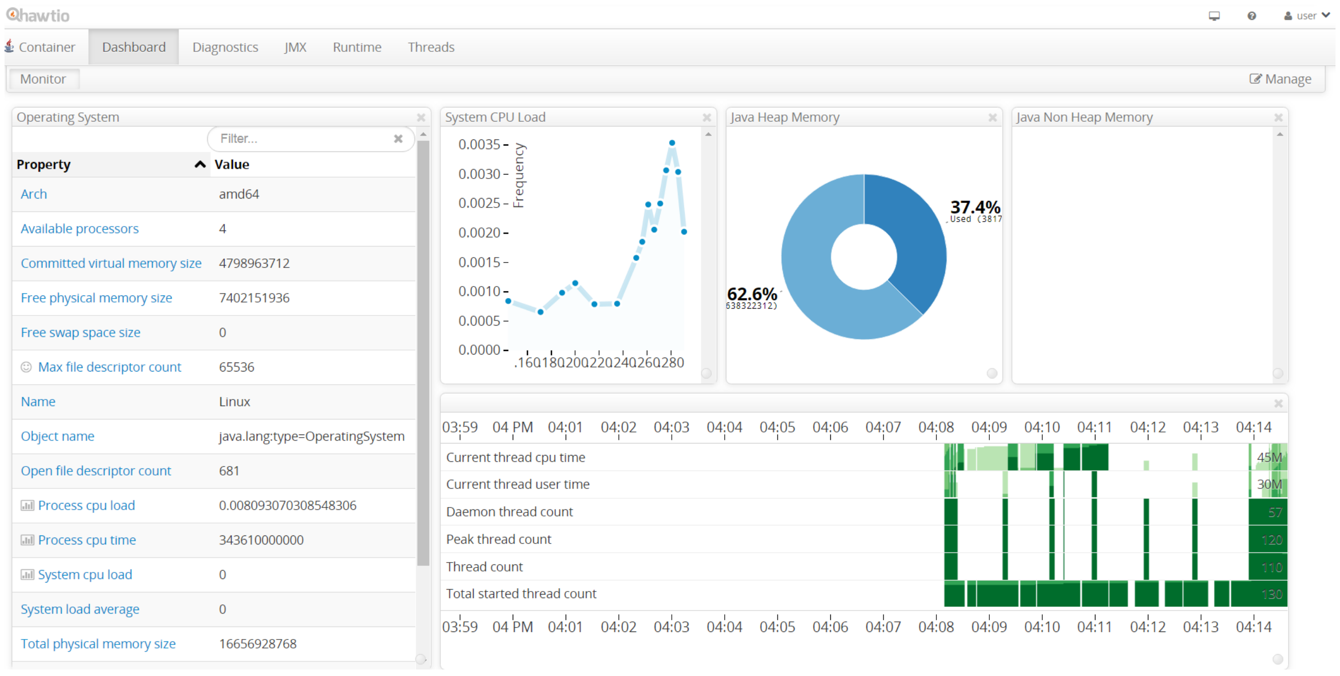
Connection Settings

Name:	Mule Dev
Scheme:	http
Host:	localhost
Port:	8899
Path:	jolokia

Figure 1. Jolokia connection settings

If your Mule instance isn't local, be sure to enter the Domain Name or IP address and make sure port 8899 is open on the Mule host.

Now for the fun part, when your connection request succeeds, you'll get a new brwoser display like this:



Property	Value
Arch	amd64
Available processors	4
Committed virtual memory size	4798963712
Free physical memory size	7402151936
Free swap space size	0
Max file descriptor count	65536
Name	Linux
Object name	java.lang:type=OperatingSystem
Open file descriptor count	681
Process cpu load	0.008093070308548306
Process cpu time	343610000000
System cpu load	0
System load average	0
Total physical memory size	16656928768

Figure 2. Hawtio Main screen

Hawtio supports various plugins that you can add to the application. Feel free to explore the various menu options and get familiar with all the capabilities.

I hope you enjoyed reading this article as much as I have enjoyed writing it, i'm looking forward to your comments!

About the Author:

Mitch Dresdner is a Senior Mule Consultant at TerraThink