MTF - monitoring via JMX and Jolokia

The goal is to be able to monitor MTF servers remotely via JMX or Jolokia (http webservices). This will allow us to use third party tools (in our case Zabbix) to monitor various aspects of the JVM such as heap usage/garbage collection etc. as well as any custom application specific mbeans we expose on the MTF server. Today, MTF does not automatically expose itself remotely via JMX. This can be done with command line arguments but would require every MTF task configuration to specify them accordingly for example you can add these following arguments:

```
java
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=12345
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
```

In a production scenario you would also probably want to enable authentication/ssl etc. We want to remove this from having to be configured though. So below I'll discuss potential approaches.

Accessing standard JMX data remotely

Option 1: Register JMX similar to what MOCA is doing

This would require a rmi-port to be specified as an argument to the MTF server (or implicitly doing this such as mtf port + 1) and we would programatically register the mbean server against this port. Add an interceptor to check credentials against either credentials configured in the MTF properties file or possibly against MOCA's console credentials (not sure about this currently)

Pros:

- JMX can be accessed remotely (connect with VisualVM/JConsole or monitoring with Zabbix via standard JMX)

Cons:

- Requires the use of another port, port either has to be set as a runtime argument or we do it implicitly with mtf port + 1
- JMX API isn't as efficient for monitoring as Jolokia bulk requests
- Lack of Jolokia means JMX information can't be accessed via http web services

Option 2: Register JMX as a seperate endpoint on the same RMI Port that Moca is using

Currently MTF runs as a Daemon task launched by MOCA. MOCA already has a defined RMI Port which is used to register the MBean server inside Moca against the endpoint "admin" i.e. it can be accessed like so:

```
service:jmx:rmi:///jndi/rmi://localhost:4510/admin
```

You can register other MBean servers from other JVMs to this same RMI port. So, technically we could register the MTF server running on port 4520 under that same RMI port like so:

```
service:jmx:rmi:///jndi/rmi://localhost:4510/mtf/4520
```

So this would consolidate all the MBean servers on one port. The MTF servers would have unique end point names like so "mtf/<telnet_port>"

Pros:

- No configuration needed on the MTF side
- JMX can be accessed remotely (connect with VisualVM/JConsole or monitoring with Zabbix via standard JMX)
- Low overhead

Cons:

- Have to be careful that end points are indeed unique
- Potentially fragile, have to make sure if Moca crashes at we have a shutdown hook to unregister itself or at least handle the scenario where the end point already exists and we unregister and register it with the new Mbean server.
- JMX API isn't as efficient for monitoring as Jolokia bulk requests
- Lack of Jolokia means JMX information can't be accessed via http web services

Accessing JMX data via Jolokia

Option 1: JMX via Jolokia

Another option would to use Jolokia via a Jolokia agent. We are doing this in MOCA but it is simple to do in MOCA because MOCA is already a web server so it's just a matter of deploying another WAR for Jolokia. In the case of MTF we are not already inside a web container. The options for Jolokia would to be either use the Jolokia JVM agent or programatically achieve what the JVM agent is doing so it doesn't require run time arguments, or potentially embed a lightweight web server like Jetty for the purpose of deploying Jolokia on MTF. A port would need to be configured to use for http access.

Pros:

- Jolokia provides web services and bulk requests, the best choice for monitoring

Cons:

- Essentially turning MTF into a web server, not sure about the resource overhead of this or any security concerns
- Requires a port to be configured
- Doesn't solve remote access via VisualVM/JConsole, still probably need to implement option 1 or 2

Option 2: JMX via Jolokia Proxy on MOCA

Another option because MOCA is already running Jolokia is that we could access MTF servers on the same machine via the Jolokia Proxy. This would allow us to make requests against the MOCA instance's deployed Jolokia servlet that would in turn make requests to MTF's MBean server.

Pros:

- Performance is still better than JMX API due to requests being fulfilled via the Jolokia servlet on the same machine rather than network round trips for each request.
- Low overhead

Cons:

- Fragile, requires that the Moca instance is running to access information about the MTF server (technically the MTF server can be running while MOCA is down due to it being a Daemon task)
- Requires the port on which to access the MBean server of the MTF server is already known (so requires option #1 or 2 to be implemented so that we know the explicit port to go to)
- Would require modifications to the Zabbix Java Gateway as we added support for Jolokia but not via Jolokia proxy.

Conclusion

The approach currently in mind is the following:

1. For standard JMX access - use Option 2 - piggy back off the Moca RMI port to expose the MTF server via standard JMX. Add an authentication interceptor, allow the user/pass to be configured in the mtf server config file
2. For Jolokia - use Option 1 - register the Jolokia JVM agent programatically. Allow for a runtime argument to be provided in MTF to define the port